Alpha-Beta Privacy

Mödersheim, Sebastian Alexander; Viganò, Luca

Published in:
ACM Transactions on Privacy and Security

Link to article, DOI:
10.1145/3289255

Publication date:
2018

Document Version
Peer reviewed version

Alpha-Beta Privacy

SEBASTIAN MÖDERSHEIM, DTU Compute, Denmark
LUCA VIGANÒ, Department of Informatics, King's College London, UK

The formal specification of privacy goals in symbolic protocol models has proved to be not quite trivial so far. The most widely used approach in formal methods is based on the static equivalence of frames in the applied pi-calculus, basically asking whether or not the intruder is able to distinguish two given worlds. But then a subtle question emerges: how can we be sure that we have specified all pairs of worlds to properly reflect our intuitive privacy goal? To address this problem, we introduce in this paper a novel and declarative way to specify privacy goals, called (α, β)-privacy. This new approach is based on specifying two formulae α and β in first-order logic with Herbrand universes, where α reflects the intentionally released information and β includes the actual cryptographic ("technical") messages the intruder can see. Then (α, β)-privacy means that the intruder cannot derive any "non-technical" statement from β that he cannot derive from α already. We describe by a variety of examples how this notion can be used in practice. Even though (α, β)-privacy does not directly contain a notion of distinguishing between worlds, there is a close relationship to static equivalence of frames that we investigate formally. This allows us to justify (and criticize) the specifications that are currently used in verification tools, and obtain a decision procedure for a large fragment of (α, β)-privacy.

CCS Concepts: • **Security and privacy → Formal security models**; **Logic and verification**; **Privacy-preserving protocols**; *Security requirements*; *Security protocols*;

Additional Key Words and Phrases: Privacy, frames, static equivalence, voting, model theory, Herbrand logic

ACM Reference Format:
Sebastian Mödersheim and Luca Viganò. 2018. Alpha-Beta Privacy. *ACM Trans. Priv. Sec.* 1, 1, Article 1 (January 2018), 33 pages. https://doi.org/10.1145/3289255

1 INTRODUCTION

1.1 Context and motivation.

Over the last fifteen years or so, several formal notions of privacy in symbolic protocol models have been proposed, e.g., [? ? ? ? ? ? ? ? ?] to name only a few. Although these notions are quite different, they are all witness to the fact that defining privacy is actually surprisingly subtle and not as easy as it is sometimes thought to be. One of the main reasons is that classical secrecy notions apply only to data that are themselves the secrets, e.g., a private key. In contrast, the data that privacy goals are formulated about are typically not secrets in themselves, e.g., the name of the candidates and of the voters in a voting protocol are usually all publicly known. Rather, the information we would like to protect is the *relation* between these values, i.e., who voted for whom.

For this reason, the majority of the popular approaches to formalizing privacy are based *not* on the question of what the intruder can deduce from a set of known messages, but rather whether

Authors' addresses: Sebastian Mödersheim, DTU Compute, Richard Petersens Plads, Building 324, Kgs. Lyngby, DK-2800, Denmark, samo@dtu.dk; Luca Viganò, Department of Informatics, King's College London, Bush House, 30 Aldwych, London, WC2B 4BG, UK, luca.vigano@kcl.ac.uk.

2471-2566/2018/1-ART1 $15.00
https://doi.org/10.1145/3289255

he can *distinguish* two different worlds.[1] A crucial question is thus: what is the "right" set of distinguishability questions to define privacy? For instance, in voting protocols, it has become standard to define vote privacy by the following "vote swapping" encoding: take the protocol with fixed votes and consider a variant where the votes of two honest voters have been swapped; then the two variants should be indistinguishable for the intruder. We propose that this vote swap, though formally precise, is still an encoding of an unspoken underlying idea or intuition. This becomes apparent when we ask the question: is vote swapping in some sense a "correct" (sound and complete) encoding? That is, does it really capture all privacy violations that we intuitively would see as a violation and only those? In some sense we cannot completely avoid the fact that there may be a gap between intuitive ideas and mathematical notions, but simple, declarative, logical notions can give a higher confidence. Below we argue that with (α, β)-privacy there is a rather straightforward formal goal we can state: that in a voting protocol the intruder only learns the number of cast votes and the result, i.e., the information that is officially published anyway. We can then prove that this is in fact equivalent to the vote-swapping trick and thereby *justify* this notion as being sound and complete with respect to a different, more high-level notion. In fact, this is the main theme of this paper: we do not want to replace the existing works on privacy, but we want to highlight it from a new angle that can give us novel insights and methods to reason about privacy.

It is even more difficult to come up with privacy definitions for other, more open-ended fields like identity management. In general, we have no good criterion when a given set of distinguishabilities is "complete", i.e., to be sure that we have not overlooked some possible connection the intruder could make that would violate our intuitive understanding of privacy. We return to this in the next subsection, after having introduced (α, β)-privacy.

1.2 Contributions.

In this paper, we take a step back and approach the problem of defining privacy from a different angle. Our overall contribution is the definition of a formal description that reflects the idea of privacy in a "natural" and less "technical" way.[2] This allows us to formally relate such declarative logical privacy definitions with existing low-level encodings based on indistinguishability: either proving that the encoding is correct or giving a counter example (e.g., where the declarative notion of privacy is violated while the encoding does not detect it). This also allows us to use existing verification technologies, but opens a new way of understanding and declaratively using the goals, and ultimately also new ways of implementing verification tools.

The logical notion we introduce in this paper is called (α, β)-privacy and it is based on specifying two formulae α and β in First-Order Logic with Herbrand Universes [**?**].

The formula α formalizes the intentionally released information, i.e., the information that we consciously give to the intruder; we also refer to this information as *payload*. This is in fact inspired by cryptographic zero-knowledge proofs: here a prover proves a *statement* to the verifier (for instance, that one possesses a credential and is over 21 years old according to this credential). The verifier clearly learns this statement, including any *logical consequence* (for instance, that the prover is also over 18 years old). The point about *zero* knowledge is that the verifier does not learn any more information than the proved statement (for instance, whether or not the prover is over

[1] There are also approaches, such as k-anonymity, ℓ-diversity, t-closeness and differential privacy, that, instead of focusing on distinguishability, aim at quantifying privacy in order to capture privacy loss and thus analyse the minimal information disclosure inherent in a system. We will discuss these approaches, and their relationship to (α, β)-privacy in more detail at the end of the paper (Section 8).

[2] Note that, as will become clearer below, when we write "less technical" here we do not mean "less formal"; rather, we mean that our definition is declarative and not at the technical level like in the case of the different indistinguishability questions where one needs to consider explicitly the cryptographic messages that are being exchanged.

65 years old). Our idea is to use such a simple formula α that describes the deliberately released information as the core of the privacy definition: the intruder should not be able to learn anything that is not a logical consequence of α.

As a counterpart to the "ideal knowledge" provided by the payload α, we also describe the *technical information* β, which represents the "actual knowledge" that the intruder has, describing the information (e.g., names, keys ...) that he initially knows, which actual cryptographic messages he has observed and what he knows about these messages. For instance, he may be unable to decrypt a message but anyway know that it has a certain format and contains certain (protected) information, like a vote.

(α, β)-privacy then means that every "non-technical" statement that the intruder can derive from β can already be derived from α. We believe that this is indeed a simple way to define privacy, and is a more declarative way to talk about privacy than distinguishability of frames. To some extent, this liberates the modelers from thinking about what technical information the intruder could exploit, and rather think about only two crucial aspects of the system that they should be clear about anyway: namely, what information is deliberately released (α) and what messages are actually exchanged (β).

For the example of vote privacy, the deliberately released information may be for instance the number of cast votes and for each candidate the number of received votes. We can then formally prove that this is indeed equivalent to the aforementioned vote-swapping trick. In other words, one can convince oneself by examples that the vote-swapping encoding makes sense: for instance, the intruder should not find out whether two honest voters have voted the same or differently, and if this were violated in a protocol, then this would be detected by the vote-swapping goal. However checking that a number of intuitive examples are covered is not completely satisfactory from a scientific point of view. To put it differently: the vote swap is an encoding of an (otherwise unformalized) intuition. With α-β privacy, in contrast, we are able to formally *prove* that it is a *correct* encoding of a simple privacy goal α: that the intruder does not find out more than we deliberately release, i.e., the number of cast votes, the election result and what the dishonest voters voted for (if that is considered).

An interesting property of (α, β)-privacy is that it is stable under background information, in the following sense: if (α, β)-privacy holds for a particular α and β, and the intruder somehow has some additional background knowledge α_0, then also $(\alpha \wedge \alpha_0, \beta \wedge \alpha_0)$-privacy holds, i.e., the intruder cannot find out more than what follows from his background knowledge and the information that was released to him.

Another interesting and declarative feature of our approach is how we can reason about the collusion of dishonest parties. If the knowledge of every dishonest party is described by an (α, β) pair, say (α_1, β_1) and (α_2, β_2), then we can consider the conjunction of the α's and the conjunction of the β's and require $(\alpha_1 \wedge \alpha_2, \beta_1 \wedge \beta_2)$-privacy to hold. This is indeed the strongest possible privacy goal any system can fulfill in the face of such a collusion, because we cannot prevent the dishonest parties from pooling their knowledge—both on the α and on the β level—and drawing conclusions from that.

Last but not least, (α, β)-privacy also provides a *model-theoretic* approach to privacy: given some information or observations about the world (a formula), what are the possible worlds that are compatible with this information (the models of the formula)? A privacy goal then specifies simply the set of possible worlds (the models of α), and the intruder should not be able to exclude any of them using β. In fact, we will show that combining the model-theoretic and proof-theoretic views yields a powerful tool-set for reasoning about privacy.

This paper is focused on developing new foundations for reasoning about privacy and we do not discuss automated procedures. However, we prove that a fragment of (α, β)-privacy corresponds to

Table 1. Roadmap of the main notions and primitives introduced

Primitive	Section	Meaning
$\Sigma, \mathcal{V}, \mathcal{T}_\Sigma(\mathcal{V})$	Section 2.1, p.4 (Definition 2.1)	Finite alphabet, disjoint set of variables, and terms of our Herbrand Logic (FOL with Herbrand Universes)
Σ_{op}	Table 2, p.8	Example set of standard cryptographic constructors, destructors, verifiers
F_i	Section 2.3, p.7 (Definition 2.6)	Frame (as in static equivalence), adapted to Herbrand Logic
m_i	Section 2.3, p.7	Memory location i, storing a piece of intruder knowledge
α	Section 3, p.10	*Payload*, information that the intruder may legitimately obtain, over $\mathcal{V}_0 \subseteq \mathcal{V}$ and $\Sigma_0 \subseteq \Sigma$
β	Section 3, p.10	*Technical information* of and about observed protocol messages, over $\mathcal{V}$ and Σ
$\phi_{frame}(F)$, $\phi_{F_1 \sim F_2}$	Fig. 1, p.9	Axioms used in (α, β)-privacy
concr	Section 3.2, p.12	Encoding of concrete intruder knowledge, ground terms from $\mathcal{T}_\Sigma$
struct	Section 3.2, p.12	Encoding of structural intruder knowledge, terms from $\mathcal{T}_\Sigma(\mathcal{V})$

frame equivalence problems, so that existing decision procedures are applicable. (α, β)-privacy is, however, more powerful and can incorporate how information is released during a protocol run. We will finally illustrate with an example how to define transition systems based on (α, β)-privacy, i.e., where every reachable state characterizes the intruder knowledge by an (α, β) pair.

1.3 Organization.

Section 2 provides the basis for our approach: we discuss First-Order Logic with Herbrand Universes, messages and frames. In Section 3, we formalize (α, β)-privacy and consider some concrete examples. Note that we introduce the main notions and primitives of our new (α, β)-privacy approach step by step, where Table 1 gives an overview of where they are introduced. In Section 4, we discuss automation and the relation of (α, β)-privacy to static equivalence. In Section 5, we provide additional examples of how (α, β)-privacy may be employed to model randomized and deterministic pooling of knowledge, and e-voting. In Section 6, we discuss how we can extend (α, β)-privacy to transition systems. In Section 7, we show how the intruder can make use of some background knowledge to carry out attacks. Finally, in Section 8, we draw conclusions, discussing also related work and future work. This paper extends and supersedes the preliminary version [**?**].

2 PRELIMINARIES

2.1 Herbrand Logic

To formalize our approach, we need to choose an appropriate logic. An obvious candidate is *first-order logic (FOL)*, but this has one difficulty when it comes to the interpretation of the constants and cryptographic operators. As is standard in security protocol verification, we would like to interpret these operators either in the free algebra or in the initial algebra induced by a set of algebraic equations; we call this the *Herbrand Universe*.[3] In general, we cannot enforce the desired

[3]Note that it is common to define the Herbrand Universe as the free term algebra but for our purposes it is crucial to include also algebraic properties of the operators, as illustrated in Example 2.3.

interpretation by axioms in FOL (see, e.g., Example 2.5). There are some workarounds for this; for instance, [? ? ? ?] use first-order Horn theories that are inconsistent (in standard FOL) iff there is an attack in the least Herbrand model, but this construction is not possible for our work because we want to talk about deductions that hold in all Herbrand models of a formula (which does not necessarily have a unique least Herbrand model).

As proposed in [?], FOL with Herbrand universes, or *Herbrand Logic* for short, can be seen as a logic in its own right—as justified by the higher expressiveness, see, e.g., Example 2.5 below. We define *Herbrand Logic* as follows (and will then discuss differences with respect to the definition of [?] below).

Definition 2.1 (Syntax of Herbrand Logic). Let $\Sigma = \Sigma_f \uplus \Sigma_i \uplus \Sigma_r$ be an alphabet that consists of

- a set Σ_f of *free function symbols*,
- a set Σ_i of *interpreted function symbols* and
- a set Σ_r of *relation symbols*,

all with their arities. We write f^n and r^n to denote a function symbol f and a relation symbol r of arity n, respectively.

We write $\mathsf{f}(t_1, \ldots, t_n)$ when $\mathsf{f} \in \Sigma_f$ and $f[t_1, \ldots, t_n]$ when $f \in \Sigma_i$, and we denote the set of considered *cryptographic operators* by the subset $\Sigma_{op} \subseteq \Sigma_f$. *Constants* are the special case of function symbols with arity 0; for an uninterpreted constant $c^0 \in \Sigma_f$, we omit the parentheses and write simply c instead of $c()$, whereas for interpreted constants $c^0 \in \Sigma_i$, we do not omit the square brackets for clarity and write $c[]$.

Let $\mathcal{V}$ be a countable set of *variable symbols*, disjoint from Σ. We denote with $\mathcal{T}_\Sigma(\mathcal{V})$ the set of all *terms* that can be built from the function symbols in Σ and the variables in $\mathcal{V}$. We simply write $\mathcal{T}_\Sigma$ when $\mathcal{V} = \emptyset$, and call its elements *ground terms* (over signature Σ). We define *substitutions* θ as is standard.

We define the set $\mathcal{L}_\Sigma(\mathcal{V})$ of *formulae* over the alphabet Σ and the variables $\mathcal{V}$ as usual: relations and equality of terms are *atomic formulae*, and *composed formulae* are built using conjunction $\wedge$, negation $\neg$, and existential quantification $\exists$.

The function *fv* returns the set of *free variables* of a formula as expected. □

We employ the standard syntactic sugar and write, for example, $\forall x.\, \phi$ for $\neg\exists x.\, \neg\phi$. We also write $x \in \{t_1, \ldots, t_n\}$ to abbreviate $x = t_1 \vee \ldots \vee x = t_n$.

Slightly abusing notation, we will also consider a substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ as a formula $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$.

Definition 2.2 (Herbrand Universe and Algebra). Formulae in Herbrand logic are always interpreted with respect to a given fixed set Σ_f of free symbols (since this set may contain symbols that do not occur in the formulae) and a congruence relation $\approx$ on $\mathcal{T}_{\Sigma_f}$. We may annotate all notions of the semantics with Σ_f and $\approx$ when it is not clear from the context.

We write $[\![t]\!]_\approx = \{t' \in \mathcal{T}_{\Sigma_f} \mid t \approx t'\}$ to denote the *equivalence class* of a term $t \in \mathcal{T}_{\Sigma_f}$ with respect to $\approx$. Further, let $U = \{[\![t]\!]_\approx \mid t \in \mathcal{T}_{\Sigma_f}\}$ be the set of all equivalence classes. We call U the *Herbrand universe* (since it is freely generated by the function symbols of Σ_f modulo $\approx$). Based on U, we define a Σ_f*-algebra* $\mathcal{A}$ that interprets every n-ary function symbol $\mathsf{f} \in \Sigma_f$ as a function $\mathsf{f}^{\mathcal{A}} : U^n \to U$ in the following standard way. $\mathsf{f}^{\mathcal{A}}([\![t_1]\!]_\approx, \ldots, [\![t_n]\!]_\approx) = [\![\mathsf{f}(t_1, \ldots, t_n)]\!]_\approx$, where the choice of the representatives $t_1, \ldots, t_n$ of the equivalence classes is irrelevant because $\approx$ is congruent. $\mathcal{A}$ is sometimes also called the *quotient algebra* (in the literature sometimes denoted with $\mathcal{T}_{\Sigma_f}/\approx$). □

Example 2.3. As an example, suppose the congruence relation $\approx$ is given by a set of equations like $\forall x, y.\, x{+}y \approx y{+}x$ for some binary function symbol $+$ in Σ_f. Then we have in the quotient

algebra 5+3 ≈ 3+5 but still 3+5 ≉ (1+2)+5. Thus, the quotient algebra is the *finest* (or "free-est") interpretation still compatible with the given algebraic properties. □

Definition 2.4 (Semantics of Herbrand Logic). An *interpretation* $\mathcal{I}$ maps every interpreted function symbol $f \in \Sigma_i$ of arity n to a function $\mathcal{I}(f) : U^n \to U$ on the Herbrand universe, every relation symbol $r \in \Sigma_r$ of arity n to a relation $\mathcal{I}(r) \subseteq U^n$ on the Herbrand universe, and every variable $x \in \mathcal{V}$ to an element of U.

We extend $\mathcal{I}$ to a function on $\mathcal{T}_\Sigma(\mathcal{V})$ as expected: $\mathcal{I}(\mathsf{f}(t_1,\ldots,t_n)) = \mathsf{f}^{\mathcal{A}}(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n))$ for $\mathsf{f} \in \Sigma_f$ and $\mathcal{I}(\mathsf{f}[t_1,\ldots,t_n]) = \mathcal{I}(\mathsf{f})(\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n))$ for $\mathsf{f} \in \Sigma_i$.

We define that $\mathcal{I}$ is a *model* of formula ϕ, in symbols $\mathcal{I} \models \phi$, as follows:

$$\begin{array}{lll} \mathcal{I} \models s = t & \text{iff} & \mathcal{I}(s) = \mathcal{I}(t) \\ \mathcal{I} \models r(t_1,\ldots,t_n) & \text{iff} & (\mathcal{I}(t_1),\ldots,\mathcal{I}(t_n)) \in \mathcal{I}(r) \\ \mathcal{I} \models \phi \wedge \psi & \text{iff} & \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \psi \\ \mathcal{I} \models \neg\phi & \text{iff} & \text{not } \mathcal{I} \models \phi \\ \mathcal{I} \models \exists x.\, \phi & \text{iff} & \text{there is a } c \in U \text{ such that } \mathcal{I}\{x \mapsto c\} \models \phi \end{array}$$

where $\mathcal{I}\{x \mapsto c\}$ denotes the interpretation that is identical to $\mathcal{I}$ except that x is mapped to c. *Entailment* $\phi \models \psi$ is defined as $\mathcal{I} \models \phi$ implies $\mathcal{I} \models \psi$ for all interpretations $\mathcal{I}$. We write $\phi \equiv \psi$ when both $\phi \models \psi$ and $\psi \models \phi$. We also use $\equiv$ in the definitions of formulae. Finally, we write $Sat(\phi)$ if ϕ has a model. □

For most applications, the interpretation of the Herbrand universe modulo a congruence $\approx$ is actually syntactic sugar. For instance, when $\approx$ is induced by a set of equations, it is not difficult to see that the relation $\approx$ can be axiomatized in Herbrand logic itself.

Example 2.5. Similar to [?], we can axiomatize arithmetic in Herbrand logic; simply let $\Sigma_f = \{\mathsf{z}^0, \mathsf{s}^1\}$, representing 0 and (+1), let $\approx$ be syntactic equality on $\mathcal{T}_{\Sigma_f}$, and let $\Sigma_i = \{add^2, mult^2\}$ and $\Sigma_r = \{<\}$ with the following formula:

$$\begin{array}{rcll} \phi & \equiv & \forall x, y. & add[\mathsf{z}, y] = y \;\wedge\; add[\mathsf{s}(x), y] = add[x, \mathsf{s}(y)] \;\wedge\; mult[\mathsf{z}, y] = \mathsf{z} \;\wedge \\ & & & mult[\mathsf{s}(x), y] = add[y, mult[x, y]] \;\wedge\; x < \mathsf{s}(x) \;\wedge\; x < y \implies x < \mathsf{s}(y) \end{array}$$

Then $\phi \models \psi$ iff ψ is a true arithmetic statement. It is well-known that (as a consequence of Löwenheim-Skolem's theorem, see [?], for instance) an equivalent axiomatization cannot be achieved in standard FOL. □

We note the following three differences with respect to the definition of Herbrand logic in [?]. First, in [?] and as is standard, the Herbrand universe is the free term algebra, forbidding one to model algebraic properties of the free operators. Our definition is a generalization to equivalence classes modulo the $\approx$ relation (and $\approx$ can simply be set to be the syntactic equality on $\mathcal{T}_{\Sigma_f}$ to get the free algebra). Second, the logic in [?] treats free variables as implicitly universally quantified, which is quite non-standard.[4] In our definition, an interpretation of a formula includes the interpretation of the free variables as is standard. This is, of course, without loss of expressiveness since one can quantify variables when this is what one wants to express. Third, the logic in [?] does not have interpreted functions and, in fact, these are syntactic sugar: an interpreted n-ary function symbol f can be modeled by an n+1-ary relation R_f symbol with the axiom $\forall x_1,\ldots,x_n.\, \exists y.\, R_f(x_1,\ldots,x_n,y) \wedge \forall y'.\, R_f(x_1,\ldots,x_n,y') \implies y = y'$.

[4] It is, of course, common to use some quantifier-free representations, e.g., a set of clauses where all variables are implicitly universally quantified. However, in a logic with (arbitrarily scoped) quantifiers this is indeed non-standard. Consider, as an example, the absurd statement $x = c \models x \neq c$. In our definition of Herbrand Logic, this is false. If we consider all variables as explicitly quantified, i.e., $\forall x.\, x = c \models \forall x.\, x \neq c$, it is trivially true (since the formula on the left is unsatisfiable).

2.2 Messages, Operators and Algebraic Properties

In (α, β)-privacy we generally use a black-box algebraic model that consists of an arbitrary set of cryptographic operators and their algebraic properties. For concreteness, for most examples in this paper, we will use the set Σ_{op} given in Table 2.

The congruence relation $\approx$ that the Herbrand universe is defined by is the least relation so that the equations in Table 2 hold (for all terms s, r, t, t_1, t_2 in $\mathcal{T}_{\Sigma_{op}}$ and for $i \in \{1, 2\}$). Intuitively,

- $\mathsf{pub}(a)$ and $\mathsf{priv}(a)$ represent an asymmetric key pair for an agent a, where pub is a public function in Σ_{op} and priv is a private function in $\Sigma \setminus \Sigma_{op}$;[5]
- $\mathsf{crypt}(p, r, t)$ represents the asymmetric encryption of a message t with a public key p and randomness r;
- $\mathsf{dcrypt}(p', t)$ represents the decryption with private key p' of a message t, and the first property formalizes that decryption with the correct private key yields the original message;
- $\mathsf{vcrypt}(p', t)$ and the second property formalize that we can check whether the message t can be correctly decrypted with private key p' (we return to this below);
- $\mathsf{sign}(p', t)$, $\mathsf{retrieve}(t)$ and $\mathsf{vsig}(p', t)$, together with their properties, similarly formalize digital signatures (where we here model signatures that contain the plaintext so that it can be retrieved);
- $\mathsf{scrypt}(k, t)$, $\mathsf{dscrypt}(k, t)$ and $\mathsf{vscrypt}(k, t)$, together with their properties, similarly formalize symmetric cryptography;
- pair, proj_i and vpair, together with their properties, formalize that we assume to have a mechanism to concatenate plaintext so that it can later be decomposed in a unique way (sometimes called "serialization");
- h is a cryptographic hash function (where the lack of destructors reflects that it is hard to find a pre-image).

This model represents cryptographic implementations that allow for checking whether one has decrypted correctly (for instance, in symmetric cryptography, this is often realized by message authentication codes). This means that the real decryption functions stop with a failure whenever the message has not been encrypted with the corresponding key, and are thus partial functions. The way we model them, destructors are total functions and the corresponding verifiers represent the domain for which they are defined. Of course, we can model other cryptographic set-ups, however they often have quite different properties for privacy, e.g., the randomization is crucial in asymmetric cryptography (as illustrated by the examples below).

Note also that our model is untyped, e.g., one can build $\mathsf{crypt}(\mathsf{h}(t_1), r, t_2)$, although decryption will later fail for any term as decryption key.

2.3 Frames

Frames and the notion of their static equivalence are a standard way to formalize privacy goals in formal methods, e.g., [**? ? ?**]. In this paper, we define frames in a slightly non-standard way that is more convenient to formalize them directly in Herbrand logic. In Section 2.4, we discuss the differences between the standard definition of frames and the one we consider here, and then, in Section 3, we relate frames to our concept of (α, β)-privacy.

Definition 2.6 (Frame). A *frame* is written as

$$F = \{\!|\mathsf{m}_1 \mapsto t_1, \ldots, \mathsf{m}_l \mapsto t_l|\!\},$$

[5]In this theory, every agent can have only one key pair for simplicity; to allow for arbitrary key infrastructures, one can rather model both pub and priv as public functions that map secret seeds to public and private keys, respectively.

Table 2. Example set Σ_{op}: standard cryptographic constructors, destructors, verifiers

Constructors	Destructors	Verifiers	Properties
pub			
crypt	dcrypt	vcrypt	$\mathsf{dcrypt}(\mathsf{priv}(s), \mathsf{crypt}(\mathsf{pub}(s), r, t)) \approx t$ $\mathsf{vcrypt}(\mathsf{priv}(s), \mathsf{crypt}(\mathsf{pub}(s), r, t)) \approx \mathsf{yes}$
sign	retrieve	vsig	$\mathsf{retrieve}(\mathsf{sign}(\mathsf{priv}(s), t)) \approx t$ $\mathsf{vsig}(\mathsf{pub}(s), \mathsf{sign}(\mathsf{priv}(s), t)) \approx \mathsf{yes}$
scrypt	dscrypt	vscrypt	$\mathsf{dscrypt}(k, \mathsf{scrypt}(k, t)) \approx t$ $\mathsf{vscrypt}(k, \mathsf{scrypt}(k, t)) \approx \mathsf{yes}$
pair	proj_i	vpair	$\mathsf{proj}_i(\mathsf{pair}(t_1, t_2)) \approx t_i$ $\mathsf{vpair}(\mathsf{pair}(t_1, t_2)) \approx \mathsf{yes}$
h			

where the m_i are distinguished constants and the t_i are ground terms that do not contain any m_i. We call $\mathsf{m}_1, \ldots, \mathsf{m}_l$ and $t_1, \ldots, t_l$ the *domain* and the *image* of the frame, respectively. □

This frame represents that the intruder *knows* l messages $t_1, \ldots, t_l$ that he can "refer to" as $\mathsf{m}_1, \ldots, \mathsf{m}_l$. In standard Dolev-Yao-style intruder models, the intruder knowledge is just a set of messages $\{t_1, \ldots, t_l\}$; in contrast, frames give each message a unique label m_i. This allows us to talk more precisely about what operations the intruder performs, e.g., "the intruder hashes the message at label m_1 and compares it with the message at label m_2". We may thus refer to the m_i as *memory locations* of the intruder knowledge.[6]

Definition 2.7 (Recipes and intruder-generable term). The set of *recipes* is the least set that contains $\mathsf{m}_1, \ldots, \mathsf{m}_l$ and that is closed under all the cryptographic operators Σ_{op}. A frame $\mathcal{F}$ can be regarded as a substitution that replaces every m_i of its domain with the corresponding t_i. For a recipe t, we thus write $\mathcal{F}(t)$ for the term obtained by applying this substitution to t. An *intruder generable term* (or just *generable term* for short) is any term s for which there is a recipe t with $s = \mathcal{F}(t)$. □

To formalize frames and recipes (and later equivalence of frames) in Herbrand Logic, we introduce two new symbols for every distinct frame $\mathcal{F}$ that we want to talk about: an interpreted unary function symbol $kn_{\mathcal{F}}$ (for *knowledge*) and a unary predicate symbol $gen_{\mathcal{F}}$ (for *generate*). We introduce two axioms $\phi_{frame}(\mathcal{F})$ and $\phi_{\mathcal{F}_1 \sim \mathcal{F}_2}$ that are shown in Fig. 1. Both these axioms are parameterized over a given set Σ_{op} of operators; for instance an expression like $\bigvee_{\mathsf{f}^n \in \Sigma_{op}} \phi$ stands for the corresponding disjunction of the instances of the formula ϕ for every operator f of arity n of Σ_{op}.

[6]The original definition of frames uses actually variables instead of constants for the labels, so that a frame *is* a substitution. In a logical context this is however quite inconvenient, since substitutions are meta level instead of object level, and mixing them usually does not lead to great results. Using constants for memory locations instead yields a clean solution and we can easily model frames as (object-level) functions on ground terms.

For every considered frame $F = \{\!|m_1 \mapsto t_1, \ldots, m_l \mapsto t_l|\!\}$, let kn_F be an interpreted unary function symbol and gen_F be a unary predicate.

For a frame F:

$$\begin{aligned}\phi_{frame}(F) \quad \equiv \quad & (\forall x.\, gen_F(x) \iff \\ & \quad (x \in \{\mathsf{m}_1, \ldots, \mathsf{m}_l\} \vee \\ & \quad\quad \bigvee_{\mathsf{f}^n \in \Sigma_{op}} \exists x_1 \ldots x_n.\, x = \mathsf{f}(x_1, \ldots, x_n) \wedge gen_F(x_1) \wedge \ldots \wedge gen_F(x_n))) \\ & \wedge \\ & (kn_F[\mathsf{m}_1] = t_1 \wedge \ldots \wedge kn_F[\mathsf{m}_l] = t_l) \\ & \wedge \\ & \bigwedge_{\mathsf{f}^n \in \Sigma_{op}} (\forall x_1 \ldots x_n. \\ & \quad gen_F(x_1) \wedge \ldots \wedge gen_F(x_n) \implies kn_F[\mathsf{f}(x_1, \ldots, x_n)] = \mathsf{f}(kn_F[x_1], \ldots, kn_F[x_n]))\end{aligned}$$

For two frames F_1 and F_2:

$$\begin{aligned}\phi_{F_1 \sim F_2} \quad \equiv \quad & (\forall x.\, gen_{F_1}(x) \iff gen_{F_2}(x)) \\ & \wedge \\ & (\forall x, y.\, gen_{F_1(x)} \wedge gen_{F_1(y)} \implies (kn_{F_1}[x] = kn_{F_1}[y] \iff kn_{F_2}[x] = kn_{F_2}[y]))\end{aligned}$$

Fig. 1. Axioms used in (α, β)-privacy parameterized over a given set Σ_{op} of operators.

Let, for instance, $F = \{\!|\mathsf{m}_1 \mapsto t_1, \mathsf{m}_2 \mapsto t_2, \mathsf{m}_3 \mapsto t_3, \mathsf{m}_4 \mapsto t_4|\!\}$ for some terms $t_1, \ldots, t_4$; then, for our example Σ_{op} of Table 2, we have

$$\begin{aligned}\phi_{frame}(F) \quad \equiv \quad & (\forall x.\, gen_F(x) \iff \\ & \quad (x \in \{\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3, \mathsf{m}_4\} \vee \\ & \quad\quad (\exists x_1.\, x = \mathsf{priv}(x_1) \wedge gen_F(x_1)) \vee \\ & \quad\quad (\exists x_1, x_2, x_3.\, x = \mathsf{crypt}(x_1, x_2, x_3) \wedge gen_F(x_1) \wedge gen_F(x_2) \wedge gen_F(x_3)) \vee \\ & \quad\quad \ldots \\ & \quad) \\ &) \\ & \wedge \\ & (\forall x_1.\, gen_F(x_1) \implies kn_F[\mathsf{priv}(x_1)] = \mathsf{priv}(kn_F[x_1])) \\ & \wedge \\ & (\forall x_1, x_2, x_3.\, gen_F(x_1) \wedge gen_F(x_2) \wedge gen_F(x_3) \implies \\ & \quad\quad kn_F[\mathsf{crypt}(x_1, x_2, x_3)] = \mathsf{crypt}(kn_F[x_1], kn_F[x_2], kn_F[x_3])) \\ & \wedge \ldots\end{aligned}$$

The formula $\phi_{frame}(F)$ characterizes a frame in Herbrand logic as follows. The first conjunct defines the predicate gen_F to be exactly the set of recipes for the frame $F = \{\!|\mathsf{m}_1 \mapsto t_1, \ldots, \mathsf{m}_l \mapsto t_l|\!\}$; the second and third conjuncts formalize that $kn_F[t]$ is the result of applying recipe t to frame F, i.e., replacing every occurrence of a label m_i by the corresponding t_i in t. Thus, we have:

$$\begin{aligned}& \mathcal{I} \models \phi_{frame}(F) \text{ iff} \\ & \quad \mathcal{I}(gen_F) = \{[t]_\approx \mid t \in \mathcal{T}_{\Sigma_{op} \cup \{\mathsf{m}_1, \ldots, \mathsf{m}_l\}}\} \text{ and} \\ & \quad \mathcal{I}(kn_F)([t]_\approx) = [F(t)]_\approx \text{ for every } t \in \mathcal{T}_{\Sigma_{op} \cup \{\mathsf{m}_1, \ldots, \mathsf{m}_l\}}\end{aligned}$$

Example 2.8. Consider the frame (from [?]):

$$F_1 = \{\!|\mathsf{m}_1 \mapsto \mathsf{scrypt}(\mathsf{k}, \mathsf{n}_1), \mathsf{m}_2 \mapsto \mathsf{pair}(\mathsf{n}_1, \mathsf{n}_2), \mathsf{m}_3 \mapsto \mathsf{k}|\!\}\,.$$

The intruder can then, e.g., obtain n_1 as follows: let $\Phi \equiv \phi_{frame}(F_1)$; then, $\Phi \models gen_{F_1}(\mathsf{dscrypt}(\mathsf{m}_3, \mathsf{m}_1)) \wedge kn_{F_1}[\mathsf{dscrypt}(\mathsf{m}_3, \mathsf{m}_1)] = \mathsf{n}_1$. We then also have $\Phi \models kn_{F_1}[\mathsf{dscrypt}(\mathsf{m}_3, \mathsf{m}_1)] = kn_{F_1}[\mathsf{proj}_1(\mathsf{m}_2)]$, which intuitively means that the intruder can check that the decrypted term is equal to the first component of the term in m_2. □

The main idea behind static equivalence of frames (e.g., [? ? ?]) is to ask whether the intruder is able to detect the difference between two "intruder knowledges" (that have the same domain): can the intruder make any check on the knowledge, i.e., two recipes so that one frame yields the same message for both recipes, while the other frame yields different messages? If there is no such critical pair of recipes, i.e., if the intruder has no way to tell whether he is "in" one frame or the other, then the frames are statically equivalent. We can formalize this in Herbrand logic using the axiom $\phi_{F_1 \sim F_2}$:

Definition 2.9 (Static Equivalence of Frames). Two frames F_1 and F_2 with the same set $\{\mathsf{m}_1, \ldots \mathsf{m}_l\}$ of memory locations are *statically equivalent* (in symbols, $F_1 \sim F_2$) iff $Sat(\phi_{frame}(F_1) \wedge \phi_{frame}(F_2) \wedge \phi_{F_1 \sim F_2})$. □

Example 2.10. We can distinguish F_1 of Example 2.8 from the frame

$$F_2 = \{\!|\mathsf{m}_1 \mapsto \mathsf{scrypt}(\mathsf{k}, \mathsf{n}_3), \mathsf{m}_2 \mapsto \mathsf{pair}(\mathsf{n}_1, \mathsf{n}_2), \mathsf{m}_3 \mapsto \mathsf{k}|\!\}$$

since the check $kn_{F_2}[\mathsf{dscrypt}(\mathsf{m}_3, \mathsf{m}_1)] = kn_{F_2}[\mathsf{proj}_1(\mathsf{m}_2)]$ fails, whereas the same check succeeds for kn_{F_1}. □

2.4 Differences with Respect to the Standard Definition of Frames

The standard definition of frames is of the form $\nu n_1, \ldots, n_k.\, \theta$, where θ is a substitution from variables $x_1, \ldots, x_l$ to terms $t_1, \ldots, t_l$, respectively, and where the x_i do not occur in the t_i. (We use the distinct constants m_i instead of the variables x_i.) Further, the n_i are the *restricted names*. Intuitively, the intruder knows all names that occur freely, i.e., that are not under a restriction. Note that in the frame $\nu n.\{\!|m_1 \mapsto n|\!\}$ the intruder can produce n anyway, even though it is restricted, since he has it as a message in his knowledge. In contrast, in our notion of frames, all constants are by default unknown to the intruder; thus, public constants must be modeled in our framework by putting them into the frame explicitly. This is of course not a restriction (if the set of public constants is finite), it only may mean longer frames. While the notion of restricted names is very handy in process calculi, they do not really fit well with a logical approach, since they are a mixture between constants and variables.

3 A NEW PRIVACY MODEL: (α, β)-PRIVACY

We introduce (α, β)-privacy step by step. In Section 3.1, we introduce the distinction between payload formulae α and technical formulae β as well as the notion of *interesting consequences*. In Section 3.2, we establish the methodology to reason over such formulae. We also define what it means for α to be combinatoric and what is a message-analysis problem. In Section 3.3, we show how (α, β)-privacy extends straightforwardly to the case of dishonest agents pooling their knowledge. In Section 4, we discuss automation and the relation to static equivalence. We then discuss further examples of (α, β)-privacy in Section 5. In Section 6, we discuss how to extend the (α, β)-privacy notion to transition systems. In Section 7, we consider the presence of additional background knowledge.

3.1 Payload and Technical Information

Our model is inspired by zero-knowledge proofs for privacy (as they are used, e.g., in IBM's Idemix [?]). The following points are characteristic for such proofs:

- The prover (intentionally) conveys some information to the verifier, i.e., the statement being proved to the verifier. We call this statement the *payload* α.
- The participants also (inevitably) convey some cryptographic information (e.g., commitments, challenges, and responses) that, if the scheme is secure, do *not* reveal anything "interesting" besides α; this, of course, is the very reason why such a scheme is called zero-knowledge. We call this kind of information the *technical information* β.

The term "interesting" that we use here is often defined in the cryptography world by the fact that it is computationally easy to produce a fake transcript of zero-knowledge proofs that is statistically indistinguishable from a real transcript. Hence, whatever information could possibly be obtained from β, one may have created oneself. This kind of definition is, however, quite unhandy in logical reasoning, and it applies only to (some types of) zero-knowledge proofs.

We show that it is fortunately possible to define the term "interesting" on a logical basis that makes sense for many actual situations in which we want to talk about privacy. The key idea is that the payload α may be formulated over a restricted alphabet $\Sigma_0 \subsetneq \Sigma$, whereas the technical information β may talk about the full alphabet Σ (e.g., all cryptographic operators are part of $\Sigma \setminus \Sigma_0$). Hence, we can define (α, β)-privacy as follows.

Definition 3.1 (Interesting consequences and respect/violation of privacy). Let $\Sigma_0 \subsetneq \Sigma$. Given a payload formula $\alpha \in \mathcal{L}_{\Sigma_0}(\mathcal{V})$ and a technical formula $\beta \in \mathcal{L}_{\Sigma}(\mathcal{V})$, where $\beta \models \alpha$ and $fv(\alpha) = fv(\beta)$ and both α and β are consistent, we say that a statement $\alpha' \in \mathcal{L}_{\Sigma_0}(fv(\alpha))$ is an *interesting consequence of* β *(with respect to* α*)* if $\beta \models \alpha'$ but $\alpha \not\models \alpha'$.

We say that β *respects the privacy of* α if it has no interesting consequences, and that β *violates the privacy of* α otherwise. □

Before we move on to the discussion of privacy on messages, let us say a few more words on the intuition behind Σ_0, α, α' and β. Σ_0 is a sub-alphabet of Σ, namely the restricted alphabet over which we can define the payloads α and the interesting consequences α'. The choice of Σ_0 depends, of course, on the specific case study we are considering, and we will give a number of examples in the next subsection and in Section 5. Σ_0 allows us to write the statements α that will be revealed to the intruder and their consequences α'. As such, Σ_0 will typically contain "payload" information, i.e., the information a system actually deals with, as opposed to all cryptographic functions and communications that are used to implement the system. For instance, in a voting system, Σ_0 could contain the names of three electoral candidates a, b and c, and α could then "reveal" that the actual vote being cast is $x \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, as opposed to the specific cryptographic messages being sent in the system that use hash functions and cryptographic operators to encrypt the votes. Note also that, given the importance of Σ_0, we could actually write "(α, β)-privacy with respect to Σ_0". However, we take the liberty to write just (α, β)-privacy as it is simpler and Σ_0 is, almost always, clear from context.

We have defined the notion of an interesting consequence α' as anything the intruder may be able to derive from his knowledge β as long as it is a non-technical statement (i.e., of $\mathcal{L}_{\Sigma_0}$) and it does not follow from α alone, i.e., from what he is permitted to know anyway. This allows us to capture that the intruder may well see a few technical details, e.g., that two messages come from the same IP address, but that in itself is not very interesting or useful as long as he cannot tie that to a relevant information α'.

Another aspect of this definition is that by the information α that we gave out, also all information that can be derived from α is given out, because the best cryptographic systems cannot prevent the intruder from drawing conclusions (by making derivations from the cryptographic messages he knows). In general, the weaker α is (i.e., the less information we deliberately release to the intruder) and the stronger β is (i.e., the more information we assume the intruder might actually have), the

stronger is the notion of privacy. So, as a rule of thumb, when in doubt, one should be restrictive on α and generous on β.

In the examples below, we will see that, at least for easy systems, it is actually possible to "infer" β systematically from the definition of the system under study: β is namely defined as the conjunction of α and the instantiation of the axioms $\phi_{frame}(\mathcal{F})$ and $\phi_{\mathcal{F}_1 \sim \mathcal{F}_2}$ for the specific system, including in particular its specific set Σ_{op} of cryptographic operators and the cryptographic messages exchanged. One could even compute β from a formal system specification of the system written, e.g., in the applied pi-calculus or in an extended Alice–Bob notation such as [**?**]. In general, however, it might not be possible to systematically infer β from the system, as β may actually contain information that goes beyond the exchanged messages, such as timing information, for instance.

3.2 Privacy on Messages

The frames that we have introduced above formalize the knowledge of the intruder. The new idea is that we also model the structural information that the intruder has about messages, and that that information takes the form of a frame too, but with variables in messages. The concrete knowledge (that is modeled by frames so far) is then an instance of this structural knowledge. Thus, a key idea of (α, β)-privacy is to make explicit the fact that the intruder will usually also have information about the structure of messages and can use this for his reasoning.

For instance, the intruder may know a message $f(a, a)$, and he may know that the term is the application of the operator f to two terms, but he may not know which terms. This structural knowledge can be expressed by the term $f(x, y)$ with two variables x and y as placeholders for what the intruder cannot determine so far. If the intruder learned that the two unknown arguments are the same, then we would have the structural information $f(x, x)$.

For most part of this paper, we will call the frame for the structural information *struct* and the frame for the concrete knowledge *concr*, and for simplicity we will abuse notation and write also $struct[\cdot]$ and $concr[\cdot]$ in Herbrand logic instead of $kn_{struct}[\cdot]$ and $kn_{concr}[\cdot]$. Since *struct* and *concr* will then also have the same domain, it follows that gen_{struct} and gen_{concr} are equivalent and thus we will simply write *gen*.

The variables that occur in frame *struct* are the free variables from α, e.g., the intruder may know about an encrypted term that it contains a particular secret, although he does not know this secret.

The axiom $\phi_{frame}(\mathcal{F})$ can now be instantiated with *concr* and *struct*. This formalizes that (a) the intruder knowledge is closed under application of public operators and (b) when the intruder composes terms himself, he knows the structure of the result as far as he knows the structure of the subterms.

An interesting question is now what it means if we also instantiate the axiom $\phi_{\mathcal{F}_1 \sim \mathcal{F}_2}$ with *concr* and *struct*, i.e., $\phi_{concr \sim struct}$. This expresses that the intruder can connect knowledge of concrete terms and their structure: two recipes yield the same intruder-generable messages iff they have the same structure.

Let us now illustrate this by introducing a running example. In (α, β)-privacy, we typically consider infinite state-transition systems and we now first focus on the analysis of privacy with respect to one particular reachable state of such a system.

Example 3.2 (A simple voting example). As an example, let us consider a very simplistic toy e-voting protocol (we will discuss more realistic examples in Section 5 and Section 6). Assume that users vote by choosing values x_i from a payload alphabet $\Sigma_0 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, and that, as part of the protocol, a voting server publishes messages of the form $\mathsf{h}(x_i)$. Consider an intermediate state of an election in which only one vote $x = \mathsf{a}$ has been cast (and the final result of the election has not

yet been released by the server). Then, α and β for this state could look as follows:

$$\begin{aligned} \alpha &\equiv x \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \\ \beta &\equiv \alpha \wedge \phi_{frame}(concr) \wedge \phi_{frame}(struct) \wedge \phi_{concr \sim struct} \end{aligned}$$

for a frame $struct = \{\!|\mathsf{m}_1 \mapsto \mathsf{a}, \mathsf{m}_2 \mapsto \mathsf{b}, \mathsf{m}_3 \mapsto \mathsf{c}, \mathsf{m}_4 \mapsto \mathsf{h}(x)|\!\}$, so that $concr = \theta(struct)$ for a substitution $\theta = \{x \mapsto \mathsf{a}\}$. Then, in particular, we have that

$$\begin{aligned} &concr[\mathsf{m}_1] = struct[\mathsf{m}_1] = \mathsf{a} \wedge \\ &concr[\mathsf{m}_2] = struct[\mathsf{m}_2] = \mathsf{b} \wedge \\ &concr[\mathsf{m}_3] = struct[\mathsf{m}_3] = \mathsf{c} \wedge \\ &concr[\mathsf{m}_4] = \mathsf{h}(\mathsf{a}) \wedge struct[\mathsf{m}_4] = \mathsf{h}(x) \end{aligned}$$

The payload formula α expresses the obvious, namely, that the intruder knows that the vote is in Σ_0. The technical formula β contains α, the concrete and structural knowledge, and the ability to compare them. β thus expresses the fact that the intruder knows

- all the constants of Σ_0 (in memory locations m_1, m_2 and m_3, for which the structural information is identical to the concrete information),
- the concrete (observed) hash of the vote (in m_4) and the structural information that the vote message has the form $\mathsf{h}(x)$.

We, the modelers, can indirectly read off from the formula β "what actually happened": there is only one interpretation of the x such that $concr[\mathsf{m}_4] = struct[\mathsf{m}_4]$ holds, namely $x = \mathsf{a}$. The intruder cannot reason this way, and thus in general will not know the right interpretation of the variables, but may in some cases be able to infer it by comparing concrete and structural information he has, as is the case in this example: first, observe that, by $\phi_{frame}(concr)$, we have $concr[\mathsf{h}(\mathsf{m}_1)] = concr[\mathsf{m}_4]$; hence, by applying $\phi_{concr \sim struct}$, we get $struct[\mathsf{h}(\mathsf{m}_1)] = struct[\mathsf{m}_4]$, which, by $\phi_{frame}(struct)$, yields $\mathsf{h}(\mathsf{a}) = \mathsf{h}(x)$; thus, we conclude that the intruder can indeed find out that the vote was $x = \mathsf{a}$, meaning that β does not respect the privacy of α (with respect to $\Sigma_0 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$) and that the protocol is not safe.

Let us therefore consider a more refined protocol by adding a fixed and secret number n: the voting server now publishes messages of the form $\mathsf{h}(\mathsf{pair}(\mathsf{n}, x_i))$ for a number n known only to the server, i.e., n is a secret from $\Sigma \setminus \Sigma_0$. (Obviously, using such a fixed number, even though secret from the intruder, is a risk for guessing attacks, but we will abstract away from guessing for now and discuss it later.) Let us again consider an intermediate state of the election in which only one vote $x = \mathsf{a}$ has been cast (and the final result of the election has not yet been released by the server). Then, the α and β for this state could look as follows:

$$\begin{aligned} \alpha &\equiv x \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \\ \beta &\equiv \alpha \wedge \phi_{frame}(concr) \wedge \phi_{frame}(struct) \wedge \phi_{concr \sim struct} \end{aligned}$$

for a frame $struct = \{\!|\mathsf{m}_1 \mapsto \mathsf{a}, \mathsf{m}_2 \mapsto \mathsf{b}, \mathsf{m}_3 \mapsto \mathsf{c}, \mathsf{m}_4 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x))|\!\}$, with $concr = \theta(struct)$ for $\theta = \{x \mapsto \mathsf{a}\}$. Now, β expresses that the intruder knows the observed hash of the vote ($concr[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n}, \mathsf{a}))$ and the structural information that this message has the form $struct[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n}, x))$. Here, the intruder does not know n and thus β has several satisfying interpretations of the free variables, i.e., β has models for each $x \in \{a, b, c\}$), which represents the uncertainty of the intruder. Hence, this β does respect the privacy of α (with respect to $\Sigma_0 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$).

One would thus be tempted to say that the variant of the protocol with the fixed secret number is safe, but what happens if there are some dishonest voters who collaborate with the intruder? Let us now focus on an intermediate state of the election in which only two votes x_1 and x_2 have been cast (and the final result of the election has not yet been released by the server), where $x_1 = \mathsf{a}$ is

from a dishonest voter and $x_2 = \mathsf{b}$ from an honest one. Then, the α and β for this state could look as follows:

$$\begin{array}{rcl} \alpha & \equiv & x_1, x_2 \in \{\mathsf{a},\mathsf{b},\mathsf{c}\} \ \wedge \ x_1 = \mathsf{a} \\ \beta & \equiv & \alpha \ \wedge \ \phi_{frame}(concr) \ \wedge \ \phi_{frame}(struct) \ \wedge \ \phi_{concr \sim struct} \end{array}$$

for a frame $struct = \{\!|\mathsf{m}_1 \mapsto \mathsf{a}, \mathsf{m}_2 \mapsto \mathsf{b}, \mathsf{m}_3 \mapsto \mathsf{c}, \mathsf{m}_4 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x_1)), \mathsf{m}_5 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x_2))|\!\}$, with $concr = \theta(struct)$ for $\theta = \{x_1 \mapsto \mathsf{a}, x_2 \mapsto \mathsf{b}\}$, where now α expresses that the intruder knows that both votes are in Σ_0 and that he knows the value of the dishonest vote, whereas β expresses that the intruder not only knows the concrete observed messages $\mathsf{h}(\mathsf{pair}(\mathsf{n},\mathsf{a}))$ and $\mathsf{h}(\mathsf{pair}(\mathsf{n},\mathsf{b}))$, which are respectively in m_4 and m_5, but he also has the structural information that these messages have the form $\mathsf{h}(\mathsf{pair}(\mathsf{n}, x_i))$.

The intruder can now reason as follows: since, by $\phi_{frame}(concr)$, it holds that $concr[\mathsf{m}_4] \neq concr[\mathsf{m}_5]$ (recall that all terms are interpreted in the Herbrand universe), we have $struct[\mathsf{m}_4] \neq struct[\mathsf{m}_5]$ by $\phi_{concr \sim struct}$, so that $\mathsf{h}(\mathsf{pair}(\mathsf{n}, \mathsf{x}_1)) \neq \mathsf{h}(\mathsf{pair}(\mathsf{n}, x_2))$, and therefore $x_1 \neq x_2$ (again since terms are interpreted in the Herbrand universe). Since the intruder knows already the dishonest vote $x_1 = a$, he knows $x_2 \neq a$, and can hence derive from β the Σ_0-formula $\alpha' \equiv x_2 \in \{\mathsf{b},\mathsf{c}\}$ that does not follow from α. Thus, in this example, β does not respect the privacy of α (with respect to $\Sigma_0 = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$). Note that the intruder cannot derive more, which is—very declaratively—because β has both a model in which $x_2 = \mathsf{b}$, and one in which $x_2 = \mathsf{c}$, so the intruder was not even able to determine the choice x_2, he was only able to exclude one interpretation, namely $x_2 = \mathsf{a}$.[7]

Let us briefly also consider three further variants of the example with $\alpha \equiv x \in \{a, b, c\}$. First, if the intruder also knows n, say $concr[\mathsf{m}_6] = struct[\mathsf{m}_6] = \mathsf{n}$, then he can indeed derive $x = \mathsf{b}$, because he can verify that $\mathsf{h}(\mathsf{pair}(\mathsf{m}_6, \mathsf{m}_2))$ gives the same concrete value as m_4.

Second, if the server uses different secret nonces instead of a fixed number for all votes, i.e., $\beta \equiv \ldots\, concr[\mathsf{m}_4] = struct[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_1,\mathsf{a})) \ \wedge \ concr[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_2,\mathsf{b})) \ \wedge \ struct[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_2, x))$, then β indeed preserves the privacy of α. To see this, note that β has models with $x = \mathsf{a}$, with $x = \mathsf{b}$, and with $x = \mathsf{c}$. Therefore, every Σ_0-formula α' that follows from β also follows from α and thus β respects the privacy of α (with respect to $\Sigma_0 = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$).

Third, so far m_4 represented the vote of a dishonest agent and the intruder knew already its value $x_1 = \mathsf{a}$. Protecting the privacy of *two honest votes* can be formalized as follows (where we have two different nonces, but model them just as some fixed constants that the intruder does not know by default):

$$\begin{array}{rcl} \alpha & \equiv & x_1 \in \{\mathsf{a},\mathsf{b},\mathsf{c}\} \ \wedge \ x_2 \in \{\mathsf{a},\mathsf{b},\mathsf{c}\} \\ \beta & \equiv & \alpha \ \wedge \ \phi_{frame}(concr) \ \wedge \ \phi_{frame}(struct) \ \wedge \ \phi_{concr \sim struct} \end{array}$$

such that β includes

$$\begin{array}{ll} \ldots & concr[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_1,\mathsf{a})) \ \wedge \ struct[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_1, x_1)) \ \wedge \\ & concr[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_2,\mathsf{b}) \ \wedge \ struct[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(\mathsf{n}_2, x_2)) \end{array}$$

Here again β respects the privacy of α (with respect to $\Sigma_0 = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$) because we can find a model for each combination of values for $x_1, x_2 \in \{\mathsf{a},\mathsf{b},\mathsf{c}\}$. In contrast, if we had used the same nonce

[7] Note also that one may, of course, consider a similar use of variables for non-payload secrets, like the value n. However, since we require that α and β have the same set of free variables, one would then existentially quantify that value, e.g.,

$$\begin{array}{rcll} \beta & \equiv & \exists y. \ \ldots & concr[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(\mathsf{n},\mathsf{a})) \ \wedge \ struct[\mathsf{m}_4] = \mathsf{h}(\mathsf{pair}(y, x_1)) \ \wedge \\ & & & concr[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(\mathsf{n},\mathsf{b})) \ \wedge \ struct[\mathsf{m}_5] = \mathsf{h}(\mathsf{pair}(y, x_2)) \end{array}$$

Without the existential quantifier (if y were left free), the intruder could derive, e.g., that $y \neq \mathsf{a}$ (by generating $\mathsf{h}(\mathsf{pair}(\mathsf{m}_1, \mathsf{m}_1))$ and comparing the result with m_4). The $\exists$ thus intuitively says that we are not interested in the concrete value of y; in fact, the goal is not the protection of the nonces in the hash-values, so if they are found out, then it is *not in itself* a violation of privacy (but may lead to one).

(replacing both n_1 and n_2 with n), we would have that $concr[\mathsf{m}_4] \neq concr[\mathsf{m}_5]$ and thus $x_1 \neq x_2$, which does not follow from α. Again the intruder does not find out x_1 or x_2 but only that the two users voted differently. The crucial point here (and the strength of (α, β)-privacy) is that we do not have to specify checks for all the different things that the intruder may be able to figure out, or even think about them; we do not need to list all the different equivalences that should be considered. In (α, β)-privacy, we simply just specify a formula α that describes what he is cleared to know and a formula β containing all information that may be available to him. □

The form of α and β that we have used for Example 3.2 is at the core of many specifications, namely, when the intruder has observed a set of messages and knows their structure. For this reason, we define a particular fragment of (α, β)-privacy (for which we give some decidability results in Section 4) that deals only with what we call *combinatoric* α and only with the analysis of messages similar to the previous example.

Definition 3.3 (Combinatoric α). We call $\alpha \in \mathcal{L}_{\Sigma_0}(\mathcal{V})$ *combinatoric* if Σ_0 is finite and contains only uninterpreted constants. □

Thus, every model $\mathcal{I}$ of α maps the free variables of α to elements of the Herbrand universe induced by Σ_0. For each free variable x of α, we have $\mathcal{I}(x) = [c]_\approx$ for some unique $c \in \Sigma_0$. For every $\mathcal{I}$ such that $\mathcal{I} \models \alpha$, we define the substitution $\theta_\mathcal{I}$ that has as domain the set of free variables of α, and such that $\theta_\mathcal{I}(x) = c$ iff $\mathcal{I}(x) = [c]_\approx$ (note that $\theta_\mathcal{I}$ is unique modulo $\approx$). Recall that, by slight abuse of notation, we may treat a substitution $\theta = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ as the Herbrand formula $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$. Thus α is equivalent to the disjunction of all such substitutions:

Lemma 3.4. *For every combinatoric α, there is a finite set of substitutions Θ such that $\alpha \equiv \bigvee_{\theta \in \Theta} \theta$. We thus call Θ also the* models *of α.*

Proof. Let $\Theta = \{\theta_\mathcal{I} \mid \mathcal{I} \models \alpha\}$. Then Θ is finite since both the domain (namely the free variables of α) and co-domain (namely Σ_0) of every $\theta_\mathcal{I}$ are finite. For every $\mathcal{I} \models \alpha$, it is clear that $\mathcal{I} \models \theta_\mathcal{I}$, thus $\alpha \models \bigvee_{\theta \in \Theta} \theta$. For every $\theta \in \Theta$ and every $\mathcal{I} \models \theta$, we have $\mathcal{I} \models \alpha$, since Σ_0 contains nothing but uninterpreted constants and all free variables of α are mapped to Σ_0 by θ, thus also $\bigvee_{\theta \in \Theta} \theta \models \alpha$. □

Definition 3.5 (Message-analysis problem). Let α be combinatoric, θ a model of α, $struct = \{\!|\mathsf{m}_1 \mapsto t_1, \ldots, \mathsf{m}_l \mapsto t_l|\!\}$ for some $t_1, \ldots, t_l \in \mathcal{T}_\Sigma(fv(\alpha))$, and $concr = \theta(struct)$. Define

$$MsgAna(\alpha,\ struct,\ \theta) \equiv \alpha \wedge \phi_{frame}(concr) \wedge \phi_{frame}(struct) \wedge \phi_{concr \sim struct}$$

If $\beta \equiv MsgAna(\alpha,\ struct,\ \theta)$, then we say that β is a *message-analysis problem (with respect to α, struct, and θ).* □

In general, such a β allows us to model a system where messages t_i have been exchanged that depend on some payload values $fv(\alpha)$ and the intruder has seen the concrete instantiations $\theta(t_i)$ of these messages. Typically, the intruder knowledge will contain all the values of Σ_0 but he does not know the substitution θ, i.e., how the payload variables were actually chosen from Σ_0. What he knows, however, is the structure of the terms, i.e., where these variables occur in the t_i, because this structural information is usually part of a publicly available protocol description. He can try to exploit comparisons ($\phi_{concr \sim struct}$) of concrete terms and structural information.

3.3 Pooling of Knowledge in (α, β)-privacy

If we model several dishonest parties, an interesting question is what they can achieve if they collaborate, in particular, if they pool their knowledge. (α, β)-privacy offers a particularly simple and declarative way of modeling, and reasoning about, the pooling of knowledge.

The general principle for pooling knowledge is as follows. We can describe the individual view of the world of two dishonest agents as two pairs (α_1, β_1)-privacy and (α_2, β_2)-privacy. Suppose they are message-analysis problems, i.e., $\beta_1 = \mathit{MsgAna}(\alpha_1,\ \mathit{struct}_1,\ \theta_1)$ and $\beta_2 = \mathit{MsgAna}(\alpha_2,\ \mathit{struct}_2,\ \theta_2)$, where we assume that the domains of struct_1 and struct_2 are disjoint (i.e., we do not have a name clash in the memory locations).

Thus, α_1 and α_2 is the knowledge we have deliberately released to the two dishonest agents, and β_1 and β_2 is the technical information such as exchanged messages that they could observe, respectively. Of course, we require that both (α_1, β_1)-privacy and (α_2, β_2)-privacy already hold. If the two parties collude, then they can in general derive more than their individual α_i, but it is quite "natural" to require that still $(\alpha_1 \wedge \alpha_2, \beta_1 \wedge \beta_2)$-privacy should hold. Why is it "natural"? It is obviously the strongest privacy requirement we can make: no system can prevent the intruders from combining their knowledge, i.e., both on the payload level $\alpha_1 \wedge \alpha_2$ and on the technical level $\beta_1 \wedge \beta_2$, and derive conclusions from that. The point is that the intruders should not be able to derive even more than that.

Example 3.6. Continuing in the realm of the previous examples, consider two dishonest agents who each have observed the messages around one vote x_1 and x_2, respectively:

$$\begin{array}{rcl} \alpha_1 & \equiv & x_1 \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \\ \beta_1 & \equiv & \mathit{MsgAna}(\alpha_1,\ \mathit{struct}_1,\ \theta_1) \end{array}$$

with $\mathit{struct}_1 = \{\!|\mathsf{m}_1 \mapsto \mathsf{a}, \mathsf{m}_2 \mapsto \mathsf{b}, \mathsf{m}_3 \mapsto \mathsf{c}, \mathsf{m}_4 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x_1))|\!\}$ and $\theta_1 = \{x_1 \mapsto \mathsf{a}\}$, and

$$\begin{array}{rcl} \alpha_2 & \equiv & x_2 \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \\ \beta_2 & \equiv & \mathit{MsgAna}(\alpha_2,\ \mathit{struct}_2,\ \theta_2) \end{array}$$

with $\mathit{struct}_2 = \{\!|\mathsf{m}_5 \mapsto \mathsf{a}, \mathsf{m}_6 \mapsto \mathsf{b}, \mathsf{m}_7 \mapsto \mathsf{c}, \mathsf{m}_8 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x_2))|\!\}$ and $\theta_2 = \{x_2 \mapsto \mathsf{b}\}$.

Now consider $\beta_1 \wedge \beta_2$. The question is whether it respects the privacy of $\alpha_1 \wedge \alpha_2$. By joining forces and pooling their knowledge, and reasoning exactly like we did in Example 3.2, the two dishonest agents can derive that the two votes are different, i.e., $x_1 \neq x_2$, which does not follow from $\alpha_1 \wedge \alpha_2$. This is an example of the fact that, even though a protocol may safeguard privacy against two intruders who do not collude and see only part of the messages, the two intruders may be able to break the protocol's privacy when they pool their knowledge. As in the previous examples, the problem is that the two messages in question use the same nonce; if the hash-values in m_4 and m_8 used different nonces n_1 and n_2 (instead of n), then the same derivation is not possible, and the privacy of $\alpha_1 \wedge \alpha_2$ is respected. □

Thus, in general, it makes sense to consider the case that dishonest agents can pool their knowledge and draw conclusions: no system can prevent this. From the (α, β)-privacy paradigm it is clear that the conjunction of the respective αs and the respective βs yields the best possible privacy requirement for the case of such a collusion, and this is a natural candidate to verify in a system.

3.4 Model-theoretical (α, β)-privacy

We now introduce a model-theoretical view of (α, β)-privacy that gives us additional tools to tackle some problems in a more semantical way. We then prove two theorems about the relationship between the *classical* notion of (α, β)-privacy that we have considered above and the model-theoretical notion.

Recall that in Herbrand logic an interpretation does not specify a universe (as is the case in standard first-order logic), but the universe is rather the Herbrand universe induced by the considered alphabet (typically Σ or Σ_0 in our case) and the congruence relation $\approx$. When it is

not determined by the context, we may explicitly denote the alphabet and write, e.g., "a Σ_0-interpretation".

Definition 3.7 (Model-theoretical (α,β)-privacy). Consider Σ_0 and Σ as before, a formula α over Σ_0 and a formula β over Σ such that $fv(\alpha) = fv(\beta)$, both α and β are consistent and $\beta \models \alpha$. We say that *(α,β)-privacy holds model-theoretically* iff every Σ_0-model of α can be extended to a Σ-model of β, where a Σ-interpretation $\mathcal{I}'$ is an *extension* of a Σ_0-interpretation $\mathcal{I}$ if they agree on all variables and all the interpreted function and relation symbols of Σ_0. □

Theorem 3.8. *If (α,β)-privacy holds model-theoretically, then it also holds in the classical sense. Conversely, if for every model $\mathcal{I}$ of α, there is a Σ_0-formula $\phi_{\mathcal{I}}$ that has only $\mathcal{I}$ as a model (with respect to Σ_0), then classical (α,β)-privacy implies model-theoretical (α,β)-privacy.*

Proof. First, suppose that (α,β)-privacy holds model-theoretically. Let $\mathcal{I}$ be a model of α. Then, for some extension $\mathcal{I}'$ of $\mathcal{I}$ (interpreting also the symbols in $\Sigma \setminus \Sigma_0$), we have $\mathcal{I}' \models \beta$. Let further α' be any Σ_0 formula that follows from β. Then, $\mathcal{I}' \models \alpha'$ and thus also $\mathcal{I} \models \alpha'$. Since $\mathcal{I} \models \alpha$ was arbitrary, we have that $\alpha \models \alpha'$; since α' was arbitrary, we have classical (α,β)-privacy.

Conversely, suppose that for every $\mathcal{I}$ such that $\mathcal{I} \models \alpha$ there is a Σ_0 formula $\phi_{\mathcal{I}}$ that has (with respect to Σ_0) exactly $\mathcal{I}$ as a model. Suppose, for the sake of contradiction, that (α,β)-privacy does not hold in the model-theoretical sense, i.e., let $\mathcal{I} \models \alpha$ be such that no extension $\mathcal{I}'$ to Σ is a model of β. Thus, no model of β is a model of $\phi_{\mathcal{I}}$, and then $\beta \models \neg\phi_{\mathcal{I}}$. Let thus $\alpha' \equiv \phi_{\mathcal{I}}$. Obviously, $\alpha \not\models \alpha'$, so classical (α,β)-privacy is violated, and we conclude by contradiction. □

Hence, for a combinatoric α, classical (α,β)-privacy and model-theoretical (α,β)-privacy coincide. In general, however, classical (α,β)-privacy does not imply model-theoretical (α,β)-privacy.

Theorem 3.9. *Classical (α,β)-privacy does not imply model-theoretical (α,β)-privacy.*

Proof. Let Σ_0 consist of the predicate p/1 and the uninterpreted functions z/0 and s/1, i.e., the Herbrand Universe is the natural numbers and p is a unary predicate over the natural numbers. We can represent any model of p as an ω-word $w \in \{0,1\}^\omega$, where the i-th position in the word is 1 iff $\mathrm{p}[\mathrm{s}^i(\mathrm{z})]$ holds. Let $\alpha \equiv \mathit{true}$. Let now Σ contain Σ_0 and the interpreted function +/2 and let

$$\beta \equiv (\forall x, y.\ (x{+}z = x)\ \wedge\ (x{+}\mathrm{s}(y) = \mathrm{s}(x{+}y)))\ \wedge\ (\exists t_1, t_2.\ (t_1 \neq \mathrm{z})\ \wedge\ (\forall x.\ \mathrm{p}[t_2{+}x] \leftrightarrow \mathrm{p}[t_1{+}t_2{+}x]))$$

Thus, β says that + is addition on natural numbers and that p when written as an ω-word has the form uv^ω for some $u, v \in \{0,1\}^*$, $v \neq \epsilon$. Clearly, (α,β)-privacy does not hold in the model-theoretical sense, but we now show that it holds in the classical sense.

Let α' be any formula over Σ_0, and let G be the models for p of the formula $\neg\alpha'$ represented as ω-words. Now G is an ω-regular language: this is because $\neg\alpha'$ is over Σ_0, thus can only use z, s and p besides the symbols of the logic; the meaning of z and s and the domain of p are determined by the Herbrand universe, thus $\neg\alpha'$ corresponds to an S1S formula and the set of its models forms an ω-regular language [**?**]. We can distinguish two cases. In the first case, $G = \emptyset$, then $\alpha \models \alpha'$, so (α,β)-privacy is not violated for this α'. In the second case, $G \neq \emptyset$. A non-empty ω-regular language, however, always contains a word of the form $w = uv^\omega$ for some $u, v \in \{0,1\}^*$, $v \neq \epsilon$. Then, w can be extended to a model of β, but it is also a model of $\neg\alpha'$, and thus $\beta \not\models \alpha'$. Hence, also for this α', (α,β)-privacy is not violated either. In conclusion, there is no α' over Σ_0 such that $\beta \models \alpha'$ but $\alpha \not\models \alpha'$. □

Note again that whenever all models of α can be written as Σ_0-formulae—which holds in all but quite construed cases—classical and model-theoretical (α,β)-privacy holds. This allows us to overcome the following obstacle. (α,β)-privacy asks for *any* Σ_0-formula α' that can be derived

from β but not from α. In general, there is a (countably) infinite choice for α' to consider. However, this is not the case when α is combinatoric:

Corollary 3.10. *Consider an (α, β) pair, where α is combinatoric and Θ is the set of models of α. Then, β violates the privacy of α iff $\beta \models \neg\theta$ for any $\theta \in \Theta$.*

Proof. For a combinatoric α, classical and model-theoretical privacy coincide by Theorem 3.8. Moreover, for a combinatoric α, Θ is finite by definition. The statement follows by the fact that $\beta \models \neg\theta$ is equivalent to $\neg Sat(\beta \wedge \theta)$. □

4 AUTOMATION AND THE RELATION TO STATIC EQUIVALENCE

The concept of (α, β)-privacy is very expressive, because Herbrand logic is. Considering Example 2.5, we recall that we can axiomatize arithmetic (of natural numbers) by a Herbrand formula α so that $\alpha \models \gamma$ iff γ is a true sentence of arithmetic. Let *valid* be a further nullary relation symbol in Σ_0 and $\beta \equiv \alpha \wedge (\gamma \implies valid)$; then β respects the privacy of α iff γ is a true sentence of arithmetic. Thus, in general, (α, β)-privacy (or its complement) is not even semi-decidable.

We see this expressive power as a feature, because it allows us to think about privacy without the tight corset imposed by automated methods. In this section, we explore a decidable fragment and the relation to static equivalence of frames for which many decidability results already exist. Because of its expressive power, it is no surprise that one can formulate static equivalence in Herbrand logic, and thus reduce static equivalence of frames to (α, β)-privacy.

Lemma 4.1. *Given two frames $\mathcal{F}_1$ and $\mathcal{F}_2$, we have $\mathcal{F}_1 \sim \mathcal{F}_2$ iff (α, β)-privacy holds for $\alpha \equiv x \in \{0, 1\}$ and $\beta \equiv \alpha \wedge (x = 1 \implies \phi_{frame}(\mathcal{F}_1) \wedge \phi_{frame}(\mathcal{F}_2) \wedge \phi_{\mathcal{F}_1 \sim \mathcal{F}_2})$.*

Proof. The statement follows straightforwardly by observing that every model of α can be extended to a model of $\phi_{frame}(\mathcal{F}_1) \wedge \phi_{frame}(\mathcal{F}_2) \wedge \phi_{\mathcal{F}_1 \sim \mathcal{F}_2}$ iff $\mathcal{F}_1 \sim \mathcal{F}_2$. □

The simple argument of this lemma may seem slightly unfair towards static equivalence of frames, since we are not truly using α for the high-level payload information available to the intruder, but rather considering everything as technical, and then just exploiting the expressive power of Herbrand logic. In order to show a closer relationship between static equivalence and (α, β)-privacy, we prove below that a large fragment of the static equivalence problem for frames can be encoded into the message-analysis fragment of (α, β)-privacy (cf. Definition 3.5).

Static equivalence of frames is essentially the question whether the intruder can distinguish two concrete worlds. For instance, the frames $\mathcal{F}_1$ and $\mathcal{F}_2$ in Example 2.8 and Example 2.10 represent two concrete worlds that the intruder can distinguish: $\mathcal{F}_1 \not\sim \mathcal{F}_2$. In contrast, (α, β)-privacy expresses with α all possible worlds (there may be more than two) and with β one concrete world, asking whether the intruder can exclude some of the worlds that are models of α. This, in particular, requires a distinction—that frames do not have—between high-level payload information and low-level technical information.

The fact that static equivalence problems can be somehow encoded into (α, β)-privacy is not too surprising since it simply exploits the expressiveness of Herbrand logic, as we already remarked. A deeper relation is highlighted in the following where we show that message-analysis-style problems with a clear distinction of payload and technical information can be "more directly" encoded into (α, β)-privacy.

The other direction for message-analysis problems is also possible. This is interesting and useful not just conceptually but also practically, since it allows us to use for (α, β)-privacy existing results and tools for static equivalence (in particular, for various algebraic theories).

As first step towards a relation between static equivalence and (α, β)-privacy, we show the following:

Lemma 4.2. *Let α be combinatoric, Θ be the models of α, and $\beta \equiv MsgAna(\alpha, \mathcal{F}, \theta_1)$ for some $\theta_1 \in \Theta$. Then, for every $\theta_2 \in \Theta$, we have $Sat(\theta_2 \wedge \beta)$ iff $\theta_1(\mathcal{F}) \sim \theta_2(\mathcal{F})$.*

Proof. The following statements are equivalent:

- $Sat(\theta_2 \wedge \beta)$
- $Sat(\theta_2 \wedge \alpha \wedge \phi_{frame}(\theta_1(\mathcal{F})) \wedge \phi_{frame}(\mathcal{F}) \wedge \phi_{\theta_1(\mathcal{F})\sim\mathcal{F}})$ [by Definition 3.5]
- $Sat(\theta_2 \wedge \alpha \wedge \phi_{frame}(\theta_1(\mathcal{F})) \wedge \phi_{frame}(\theta_2(\mathcal{F})) \wedge \phi_{\theta_1(\mathcal{F})\sim\theta_2(\mathcal{F})})$
  This is because the formula contains θ_2 and thus all variables must be instantiated accordingly, including those occurring in $\mathcal{F}$.
- $Sat(\phi_{frame}(\theta_1(\mathcal{F})) \wedge \phi_{frame}(\theta_2(\mathcal{F})) \wedge \phi_{\theta_1(\mathcal{F})\sim\theta_2(\mathcal{F})})$
  This is because the formula $Sat(\phi_{frame}(\theta_1(\mathcal{F})) \wedge \phi_{frame}(\theta_2(\mathcal{F})) \wedge \phi_{\theta_1(\mathcal{F})\sim\theta_2(\mathcal{F})})$ is ground (and consistent with α and θ_2).
- $\theta_1(\mathcal{F}) \sim \theta_2(\mathcal{F})$ [by Definition 2.9] □

We now show the equivalence of message-analysis problems with a corresponding finite set of static equivalence problems. More specifically, we prove that it suffices to pick arbitrarily one of the models of α and show that (α, β)-privacy holds for that model, and that this is equivalent to showing the indistinguishability of the given frame under every model of α.

Theorem 4.3. *Let α be combinatoric, $\Theta = \{\theta_1, \ldots, \theta_n\}$ be the models of α, and $\beta \equiv MsgAna(\alpha, \mathcal{F}, \theta_1)$ for some $\theta_1 \in \Theta$. Then, we have that (α, β)-privacy holds iff $\theta_1(\mathcal{F}) \sim \ldots \sim \theta_n(\mathcal{F})$.*

Proof. The following are equivalent:

- (α, β)-privacy
- $Sat(\theta_i \wedge \beta)$ for every $1 \leq i \leq n$ [by Definition 3.7]
- $\theta_i(\mathcal{F}) \sim \theta_1(\mathcal{F})$ for every $1 \leq i \leq n$ [by Lemma 4.2]
- $\theta_1(\mathcal{F}) \sim \ldots \sim \theta_n(\mathcal{F})$ [by Definition of $\sim$] □

Example 4.4. Let us consider again the second case of our simple voting example (Example 3.2), in which the voting server publishes messages of the form $\mathsf{h}(\mathsf{pair}(\mathsf{n}, x_i))$ for a fixed number n known only to the server, i.e., n is a secret from $\Sigma \setminus \Sigma_0$. The models of

$$\alpha \quad \equiv \quad x \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$$

are $\Theta = \{\theta_1, \theta_2, \theta_3\}$ with $\theta_1 = \{x \mapsto \mathsf{a}\}$, $\theta_2 = \{x \mapsto \mathsf{b}\}$ and $\theta_3 = \{x \mapsto \mathsf{c}\}$. Then, for $\theta_i \in \Theta$ corresponding to the specific vote x_i that has been cast, we have

$$\begin{aligned} \beta \quad &\equiv \quad MsgAna(\alpha, \mathit{struct}, \theta_i) \\ &\equiv \quad \alpha \wedge \phi_{frame}(\mathit{concr}) \wedge \phi_{frame}(\mathit{struct}) \wedge \phi_{\mathit{concr}\sim\mathit{struct}} \end{aligned}$$

for a frame $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \mathsf{a}, \mathsf{m}_2 \mapsto \mathsf{b}, \mathsf{m}_3 \mapsto \mathsf{c}, \mathsf{m}_4 \mapsto \mathsf{h}(\mathsf{pair}(\mathsf{n}, x))|\!\}$, with $\mathit{concr} = \theta_i(\mathit{struct})$. So, to check whether (α, β)-privacy holds it suffices to check one $\theta_i \in \Theta$, say $\theta_1 = \{x \mapsto \mathsf{a}\}$, and then it holds for all $\theta_i \in \Theta$. Theorem 4.3 tells us that this is equivalent to checking the static equivalence between all concrete votes by considering all models, i.e., $\theta_1(\mathit{struct}) \sim \theta_2(\mathit{struct}) \sim \theta_3(\mathit{struct})$. □

Corollary 4.5. *Let α be combinatoric, Θ be the models of α, and $\mathcal{F}$ be a frame. Then, we have that $(\alpha, MsgAna(\alpha, \mathcal{F}, \theta))$-privacy holds for* some *$\theta \in \Theta$ iff $(\alpha, MsgAna(\alpha, \mathcal{F}, \theta))$-privacy holds for* all *$\theta \in \Theta$.*

Hence, if in a given world θ the intruder cannot exclude any other worlds, then in no world he can exclude any world. Thus, in transition systems, we may summarize the worlds that only differ on the concrete value of the privacy variables in one state. This is however only possible as long as no behavior of the honest agents depends on it. For instance, in a voting protocol, the actual values of the vote have no influence on the behavior of the honest agents, so there we can always model

all possible voting outcomes by a single state that is parameterized over the choice of a model θ. Thus, we can exploit privacy (the goal) even for an efficient representation.

Since this result is independent of the considered set Σ_{op} of cryptographic operations and algebraic theory, we immediately have that if we can decide static equivalence for a given theory (e.g., [? ?]), then we can decide the message-analysis problem fragment of (α, β)-privacy for that theory.

To conclude the section, note that, instead of relying on static equivalence, we could have also given a direct decision procedure for our example theory, without an enumeration of all models.

5 MODELING AND REASONING ABOUT FURTHER EXAMPLE SCENARIOS

We chose the following major areas to model further examples and show (α, β)-privacy at work:

- randomized vs. non-randomized encryption including non-determinism and the notion of strong secrecy (Section 5.1),
- guessing attacks (Section 5.2, in which we discuss different approaches to encode passwords and guessing in (α, β)-privacy and show unique features of our logic), and
- e-voting (Section 5.3).

5.1 Modeling Subtleties of Encryption

5.1.1 Randomized vs. Non-Randomized Encryption. Standard Dolev-Yao models are blind to the problem of non-randomized encryption because the intruder cannot compare results in these models. Here, we have directly modeled randomized encryption operators by means of a random value r, e.g., $\mathrm{crypt}(k, r, m)$, where the randomization should, of course, be different at each encryption. In our approach, this is taken care of by the transition system, which should actually ensure that every honest agent chooses a new randomness at each encryption.

We model non-randomized encryption by using $r = \varepsilon$ for an intruder-known constant ε, i.e., we write $\mathrm{crypt}(\cdot, \varepsilon, \cdot)$ to model the missing randomization.

Example 5.1 (Simple Voting). Let $\Sigma_0 = \{0, 1\}$, $\alpha \equiv x \in \{0, 1\}$, and $\beta \equiv \mathit{MsgAna}(\alpha,\ \mathit{struct},\ \theta)$, where $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \varepsilon, \mathsf{m}_2 \mapsto 0, \mathsf{m}_3 \mapsto 1, \mathsf{m}_4 \mapsto \mathsf{k}, \mathsf{m}_5 \mapsto \mathrm{crypt}(\mathsf{k}, \varepsilon, x)|\!\}$ and $\theta = \{x \mapsto 0\}$. The intruder can derive from β that $\mathit{concr}[\mathsf{m}_5] = \mathrm{crypt}(\mathsf{k}, \varepsilon, 0) = \mathit{concr}[\mathrm{crypt}(\mathsf{m}_4, \mathsf{m}_1, \mathsf{m}_2)]$, and then, by $\phi_{F_1 \sim F_2}$, $\mathit{struct}[\mathsf{m}_5] = \mathit{struct}[\mathrm{crypt}(\mathsf{m}_4, \mathsf{m}_1, \mathsf{m}_2)]$ and thus $x = 0$.

A similar deduction would not be possible if crypt contained additionally something random, i.e., if $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \varepsilon, \mathsf{m}_2 \mapsto 0, \mathsf{m}_3 \mapsto 1, \mathsf{m}_4 \mapsto \mathsf{k}, \mathsf{m}_5 \mapsto \mathrm{crypt}(\mathsf{k}, \mathsf{r}, x)|\!\}$, where r is not known to the intruder. (This is of course the very reason for probabilistic encryption.) In such a probabilistic variant, β would indeed respect the privacy of α as the intruder would no longer be able to generate terms that would give him any interesting insight on x. □

5.1.2 Non-determinism. Another interesting question that we can ask is what the intruder can discover when he does not know the content of messages sent by the honest agents (in fact, he might not even know to which protocol the messages belong), but he knows the format of the messages that are exchanged in several protocols. (α, β)-privacy can easily encode non-determinism, e.g., if messages have a different structure depending on the choices of parties. Consider this example

$$\alpha \ \equiv\ x \in \{\mathsf{yes}, \mathsf{no}\} \ \wedge\ y \in \{\mathsf{a}, \mathsf{b}\}\,,$$

where x is the server's decision and y is the concrete name of the client (the intruder knows yes, no and pub(a)). A party may respond to a request with

$$\mathrm{crypt}(\mathrm{pub}(y), \mathsf{r}, \mathrm{pair}(\mathsf{yes}, \mathsf{n})) \quad \text{or} \quad \mathrm{crypt}(\mathrm{pub}(y), \mathsf{r}, \mathsf{no})\,,$$

depending on the decision on the vote, where n is a secret from $\Sigma \setminus \Sigma_0$ (and where, like in Example 3.2, we abstract away from the fact that the intruder might be able to guess such a fixed number). This is no longer a message-analysis problem, since the structure of the messages depends on x; and it cannot be directly expressed as a problem of static equivalence of frames either.[8] However, (α, β)-privacy allows us to incorporate such more complex relations between α and the structure of messages, e.g., the history of events that happened so far. For this example, we can specify the *struct* and *concr* knowledge of the intruder directly as a formula related to the flag x as part of β:

$$\begin{aligned} &\mathit{concr}[\mathsf{m}_1] = \mathit{struct}[\mathsf{m}_1] = \mathsf{yes} \wedge \\ &\mathit{concr}[\mathsf{m}_2] = \mathit{struct}[\mathsf{m}_2] = \mathsf{no} \wedge \\ &\mathit{concr}[\mathsf{m}_3] = \mathit{struct}[\mathsf{m}_3] = \mathsf{pub}(\mathsf{a}) \wedge \\ &\mathit{concr}[\mathsf{m}_4] = \mathsf{crypt}(\mathsf{pub}(\mathsf{a}), \mathsf{r}, \mathsf{pair}(\mathsf{yes}, \mathsf{n})) \wedge \\ &\qquad ((x = \mathsf{yes} \ \wedge \ \mathit{struct}[\mathsf{m}_4] = \mathsf{crypt}(\mathsf{pub}(y), \mathsf{r}, \mathsf{pair}(\mathsf{yes}, \mathsf{n}))) \vee \\ &\qquad\ (x = \mathsf{no} \ \wedge \ \mathit{struct}[\mathsf{m}_4] = \mathsf{crypt}(\mathsf{pub}(y), \mathsf{r}, \mathsf{no}))) \end{aligned}$$

so that β actually encodes a concrete world (in this case, such that $x = \mathsf{yes}$).

Indeed, in the example, (α, β)-privacy holds, but if we had used non-randomized encryption (replacing r with ε), the intruder could have generated the term $\mathsf{crypt}(\mathsf{m}_3, \varepsilon, \mathsf{m}_2)$ and checked that it does not produce the same *concr* value as m_4. Thus, $\beta \models \mathit{struct}[\mathsf{m}_4] \neq \mathsf{crypt}(\mathsf{pub}(\mathsf{a}), \varepsilon, \mathsf{no})$. It follows that the model $(x = \mathsf{no} \ \wedge \ y = \mathsf{a})$ is excluded. Note that this does not tell us the value of x nor that of y, but just excludes one of the combinations.

5.1.3 Strong Secrecy. In the static equivalence community (e.g., [**?**]), there is also the notion of strong secrecy:

Definition 5.2. A frame F that talks about a variable x (the "secret") *respects the strong secrecy of* x if $F\{x \mapsto s\} \sim F\{x \mapsto t\}$ for any generable terms s and t. □

Thus, if the intruder can choose arbitrary (known) values and the secret is replaced for those values, he still cannot deduce which one it is.

The philosophy of the (α, β)-privacy approach is actually a bit different from this view: we would normally consider problems of strong secrecy actually as the question of checking that a protocol provides secrecy even when the secrets are weak, like poor passwords. We illustrate this in the following subsection on modeling guessing attacks. However, the notion of a game that the intruder can play is quite interesting for its relation for instance to cryptographic models. Therefore, we quickly illustrate how this can be done with (α, β)-privacy.

Example 5.3 (Strong Secrecy Game). As an example of the experiment that the intruder can do with his environment, consider the following "strong secrecy game". To model an interaction between the intruder and a (virtual) host, we design a formula that represents the evolution of the intruder knowledge in the game in two steps:

- We start in a state where the intruder knowledge is represented by $F_1 = \{\!|\mathsf{m}_1 \mapsto \mathsf{n}_1, \mathsf{m}_2 \mapsto \mathsf{n}_2|\!\}$.
- The intruder can choose any two recipes s_0 and s_1 with respect to F_1, i.e., $gen_{F_1}(s_0) \wedge gen_{F_1}(s_1)$, and he sends $\mathit{concr}[s_0]$ and $\mathit{concr}[s_1]$ to an honest agent host.
- Now the host non-deterministically chooses a Boolean $b \in \{0, 1\}$ and encrypts $\mathit{concr}[s_b]$ with a key k and sends it back to the intruder, who only knows that the encryption contains

[8] Note that, in general, even for problems that fall outside the message-analysis fragment of (α, β)-privacy there could still be suitable encodings into static equivalence problems.

s_b depending on the choice b (that the intruder does not know). For this uncertainty, we introduce a new variable x and have $\alpha \equiv x \in \{0,1\}$. Let $\theta = [x \mapsto b]$. Now we define

$$\begin{array}{rcl}\beta & \equiv & \mathit{MsgAna}(\alpha, \Gamma_1, \theta) \\ & & \wedge \\ & & \exists s_0, s_1.\ \mathit{gen}_{\Gamma_1}(s_0) \wedge \mathit{gen}_{\Gamma_1}(s_1) \\ & & \quad \wedge \exists s_x.\ s_x \in \{s_0, s_1\}. \\ & & \quad\quad ((x = 0 \wedge \mathit{MsgAna}(\alpha, \Gamma_2, \theta \cup [s_x \mapsto s_0])) \vee \\ & & \quad\quad\ (x = 1 \wedge \mathit{MsgAna}(\alpha, \Gamma_2, \theta \cup [s_x \mapsto s_1])))\end{array}$$

where $\Gamma_2 = \Gamma_1 \cup \{\!|\mathsf{m}_3 \mapsto \mathsf{scrypt}(k, \mathit{concr}[s_x])|\!\}$.

Indeed, (α, β)-privacy holds, i.e., the intruder cannot determine x, so this gives us strong secrecy. An example of violation of strong secrecy would be for instance if the message had rather the form $\mathsf{scrypt}(\mathit{concr}[s_b], \ldots)$ (i.e., if a weak secret is used to encrypt a message) or if we have hash-values like $h(\mathit{concr}[s_b], \ldots)$ with other guessable elements, since then the intruder can verify whether the message was generated with s_0 or s_1, and thus obtain x. □

5.2 Modeling Guessing Attacks

Guessing attacks have been intensively studied in security protocol analysis, also using static equivalence, e.g., [? ? ?] to name just a few works. We can use also this example to show how flexible our (α, β)-privacy approach is: suppose that we have an intruder who knows only a part of the password space (e.g., some users use good passwords, some use bad passwords). Let $P = \{\mathsf{p}_1, \ldots, \mathsf{p}_k\}$ be the space of all passwords and let $D = \{\mathsf{p}_1, \ldots, \mathsf{p}_l\}$, with $l < k$, be the intruder's dictionary. For concreteness, let us consider the classical example of the MS CHAPv2 protocol [?], where the server sends a nonce ns and the client c should produce $\mathsf{nc}, \mathsf{h}_x(\mathsf{nc}, \mathsf{ns}, \mathsf{c})$ for some client nonce nc, hash-MAC'ed with the client's password x. Then one reachable state could be:

$$\begin{array}{rcl}\alpha & \equiv & x \in \{\mathsf{p}_1, \ldots, \mathsf{p}_k\} \\ \beta & \equiv & \mathit{MsgAna}(\alpha, \mathit{struct}, \theta)\end{array}$$

where $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \mathsf{p}_1, \mathsf{m}_2 \mapsto \mathsf{p}_2, \ldots, \mathsf{m}_l \mapsto \mathsf{p}_l, \mathsf{m}_{l+1} \mapsto \mathsf{ns}, \mathsf{m}_{l+2} \mapsto \mathsf{c}, \mathsf{m}_{l+3} \mapsto \mathsf{pair}(\mathsf{nc}, \mathsf{h}(x, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c}))))|\!\}$ and $\theta = \{x \mapsto \mathsf{p}_1\}$.

This is obviously the concrete state where $x = \mathsf{p}_1$, and the intruder can find out that that is indeed the case. To that end, he uses the generable terms $s = \mathsf{h}(\mathsf{m}_1, \mathsf{pair}(\mathsf{proj}_1(\mathsf{m}_{l+3}), \mathsf{pair}(\mathsf{m}_{l+1}, \mathsf{m}_{l+2})))$ and $t = \mathsf{proj}_2(\mathsf{m}_{l+3})$, so that $\mathit{concr}[s] = \mathit{concr}[t]$ and thus $\mathit{struct}[s] = \mathit{struct}[t]$. Then

$$\mathsf{h}(\mathsf{p}_1, \mathsf{pair}(\mathsf{proj}_1(\mathsf{pair}(\mathsf{nc}, \mathsf{h}(x, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c}))))), \mathsf{pair}(\mathsf{ns}, \mathsf{c}))) = \mathsf{proj}_2(\mathsf{pair}(\mathsf{nc}, \mathsf{h}(x, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c}))))))$$

and thus $\mathsf{h}(\mathsf{p}_1, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c}))) = \mathsf{h}(x, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c})))$. In the given Herbrand universe (which depends on the algebraic theory), the only possible model for x is $x = \mathsf{p}_1$, thus $\beta \models x = \mathsf{p}_1$.

Note that if we consider a trace where $x = \mathsf{p}_n$ with $l < n \leq k$, thus using a password that the intruder does not have in his dictionary, then this attack is not possible. However, the intruder can still derive $x \in \{\mathsf{p}_{l+1}, \ldots, \mathsf{p}_k\}$ since by checking his entire dictionary, he can confirm that the password is not one of those he knows. So, in fact, privacy is violated also in these cases, even though most people would agree that this in itself is not a problem, at least for online guessing: an interesting argument, in this case, is that even the best protocol and the best password cannot prevent the intruder from trying out online a few passwords and confirming that they are wrong guesses. Whereas in offline guessing good password protocols don't in fact allow the intruder to carry out offline guesses, we can still consider transition systems in which the intruder can actually send guesses to the server. We leave a more detailed investigation of this for future work.

In fact, for this reason we suggest to see the password itself as a technical information that encrypts the actual payload information we try to protect. This is a simple way to say: *we don't care what the intruder finds out about the password, but about the messages that are encrypted with it.* This allows us to circumvent all troubles with declassification of information or quantitative aspects. Here is how this could look like, if the password p_n is used to encrypt a payload x that is, say, from a set of values $\{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$:

$$\begin{array}{lcl} \alpha & \equiv & x \in \{\mathsf{c}_1, \ldots, \mathsf{c}_m\} \\ \beta & \equiv & \mathit{MsgAna}(\alpha,\ \mathit{struct},\ \theta) \end{array}$$

where $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \mathsf{p}_1, \mathsf{m}_2 \mapsto \mathsf{p}_2, \ldots, \mathsf{m}_l \mapsto \mathsf{p}_l, \mathsf{m}_{l+1} \mapsto \mathsf{ns}, \mathsf{m}_{l+2} \mapsto \mathsf{c}, \mathsf{m}_{l+3} \mapsto \mathsf{pair}(\mathsf{nc}, \mathsf{h}(\mathsf{p}_n, \mathsf{pair}(\mathsf{nc}, \mathsf{pair}(\mathsf{ns}, \mathsf{c})))), \mathsf{m}_{l+4} \mapsto \mathsf{scrypt}(\mathsf{p}_n, x)|\!\}$ and $\theta = \{x \mapsto \mathsf{c}_7\}$.

This preserves privacy if $l < n \leq k$ (i.e., for a good password) and still violates it for a bad password $1 \leq n \leq l$. Note that when the password is technical information, we do no longer use a variable for it, because it is not part of the space we try to get information about.

5.3 Modeling E-Voting

An interesting field for privacy goals is of course electronic voting (e-voting), which we already touched upon in the previous examples. In this paper, we do not want to model and analyze a full voting protocol, as that is not the main focus of our research, but rather only illustrate how (α, β)-privacy gives a new and declarative way to formulate privacy goals and describe their analysis.

Let us first look at the most basic setting: the vote between two options 1 and 0 (e.g., representing "yes" or "no", or the choice between two candidates).

Definition 5.4 (Binary Voting Privacy). Consider a voting system where the choice is either 1 or 0. Let N be the number of cast votes and R be the number of votes for 1. Let Σ_0 consist of the constant 0, the successor function $s(\cdot)$ and the addition function $+$, and consider the axiom α_{ax} characterizing addition:

$$\alpha_{ax} \equiv \forall x. \forall y.\ x + 0 = x \wedge x + s(y) = s(x + y)$$

The *privacy goal for binary votes* (namely releasing N and R) is then defined by the following formula α:

$$\alpha \equiv \alpha_{ax} \wedge v_1 \in \{0, 1\} \wedge \ldots \wedge v_N \in \{0, 1\} \wedge v_1 + v_2 + \ldots + v_N = R\,,$$

where $v_1, \ldots, v_N$ are variables. □

Note that, strictly speaking, α is not combinatoric, since Σ_0 contains 0 and s (so the Herbrand universe entails the natural numbers) and $+$ is interpreted as addition. There are however only finitely many models (with respect to the free variables). In fact, there exists an (inefficient) encoding into a combinatoric problem that works without a concept of natural numbers.

Example 5.5. For example, consider the trivial voting protocol, in which every voter i sends their vote v_i directly to a voting server, signed with their own private key and encrypted with the server's public key. This of course makes the strong requirement that the server must be completely trusted by everybody as it can see all votes and nobody can verify the correct tallying by the server. Let θ be the true result of the vote (mapping each v_i to 0 or 1); then we have $\beta = \mathit{MsgAna}(\alpha,\ \mathit{struct},\ \theta)$ where

$$\begin{array}{l} \mathit{struct} = \{\!|m_1 \mapsto \mathsf{crypt}(\mathsf{pub}(s), r_1, \mathsf{sign}(\mathsf{priv}(1), \mathsf{ballot}(v_1))), \\ \qquad \ldots \\ \qquad m_N \mapsto \mathsf{crypt}(\mathsf{pub}(s), r_N, \mathsf{sign}(\mathsf{priv}(N), \mathsf{ballot}(v_N)))|\!\} \end{array}$$

and ballot is the ballot format of this vote, priv(i) is the private key of the ith voter, pub(s) is the public key of the server, and the r_i are random values to make the encryption non-deterministic.

In this example, (α, β)-privacy holds, since the intruder cannot analyze any messages and cannot compare them, thus every model of α can be extended to a model of β. □

There are many works on electronic voting using the applied pi-calculus or a similar calculus to model the protocol (see, e.g., [? ? ? ? ? ?] to name a few). The voting privacy goal is usually then expressed by the following trick: consider two processes that differ only in the concrete vote of two (honest) voters who swap their vote; then any such pair of processes must be indistinguishable. More formally, let θ_1 and θ_2 be two substitutions with domain $v_1, \ldots, v_N$ and co-domain $\{0, 1\}$. Then θ_1 and θ_2 are a *vote swap* of each other, if $\theta_1(v_i) = \theta_2(v_j)$ and $\theta_1(v_j) = \theta_2(v_i)$ for some $i, j \in \{1, \ldots, N\}$ and $\theta_1(v_k) = \theta_2(v_k)$ for all other k. We show that this privacy definition based on vote swapping is actually in some sense equivalent to the binary voting privacy goal:

Theorem 5.6. *Let α be the binary voting privacy goal from Definition 5.4, θ_0 be a model of α, and $\beta \equiv MsgAna(\alpha, struct, \theta_0)$ for some frame struct be a message-analysis problem. Then (α, β)-privacy holds iff $\theta_1(struct) \sim \theta_2(struct)$ for any two models θ_1 and θ_2 of α that are a vote swap of each other.*

Proof. Let $\Theta = \{\theta_1, \ldots, \theta_n\}$ be the models of α. By Theorem 4.3, (α, β)-privacy holds iff $\theta_1(struct) \sim \ldots \sim \theta_n(struct)$. The models of α can be characterized by permutations: θ_1, θ_2 are models of α iff there is a permutation π of $\{1, \ldots, N\}$ such that $\theta_1(v_i) = \theta_2(v_{\pi(i)})$ for all $i \in \{1, \ldots, N\}$. The theorem now follows from the fact that all permutations can be obtained from each other by a sequence of swaps, and $\sim$ is an equivalence relation. □

Example 5.7. Let us consider the setting of Definition 5.4 and Example 5.5 and let, for concreteness, the number of cast votes be $N = 3$ and the number of votes for 1 be $R = 2$. Then the privacy goal for binary votes is defined by

$$\alpha \equiv \alpha_{ax} \wedge v_1 \in \{0, 1\} \wedge v_2 \in \{0, 1\} \wedge v_3 \in \{0, 1\} \wedge v_1 + v_2 + v_3 = 2\,,$$

where v_1, v_2, v_3 are variables and α_{ax} is the axiom characterizing addition. The models of α are $\Theta = \{\theta_1, \theta_2, \theta_3\}$ with $\theta_1 = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\}$, $\theta_2 = \{v_1 \mapsto 1, v_2 \mapsto 0, v_3 \mapsto 1\}$ and $\theta_3 = \{v_1 \mapsto 0, v_2 \mapsto 1, v_3 \mapsto 1\}$.

Let, without loss of generality, the true result of the vote be $\theta = \theta_1 = \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 0\}$, so that $\beta = MsgAna(\alpha, struct, \theta)$ with

$$\begin{aligned} struct = \{\!|\, & \mathsf{m}_1 \mapsto \mathsf{crypt}(\mathsf{pub}(s), r_1, \mathsf{sign}(\mathsf{priv}(1), \mathsf{ballot}(v_1))), \\ & \mathsf{m}_2 \mapsto \mathsf{crypt}(\mathsf{pub}(s), r_2, \mathsf{sign}(\mathsf{priv}(2), \mathsf{ballot}(v_2))), \\ & \mathsf{m}_3 \mapsto \mathsf{crypt}(\mathsf{pub}(s), r_3, \mathsf{sign}(\mathsf{priv}(3), \mathsf{ballot}(v_3))), \\ & \mathsf{m}_4 \mapsto 0, \mathsf{m}_5 \mapsto 1, \mathsf{m}_6 \mapsto \mathsf{pub}(s) \,|\!\} \end{aligned}$$

As we remarked above, (α, β)-privacy holds since the intruder cannot analyze any messages and cannot compare them, thus every model of α can be extended to a model of β. We can also use Theorem 5.6 (and Theorem 4.3) to argue that (α, β)-privacy holds iff $\theta_1(struct) \sim \theta_2(struct) \sim \theta_3(struct)$. It is easy to see that the models of α are characterized by permutations of $\{1, 2, 3\}$, and that all these permutations can be obtained from each other by a sequence of swaps. Hence, we can conclude by the fact that $\sim$ is an equivalence relation. □

Theorem 5.6 shows that (α, β)-privacy is actually a declarative logical characterization of the privacy goals, in contrast to the more technical vote-swapping formulation. Note also that privacy for some more advanced voting systems cannot be characterized by vote swapping, but such systems have a declarative specification using (α, β)-privacy. For example, consider the following generalization of voting privacy:

Definition 5.8 (General Vote Privacy). Consider a voting system with K candidates $\{c_1, \ldots, c_K\}$ and N voters, and such that every voter has L votes (one can give multiple votes to candidates). We use a binary interpreted function symbol $v[\cdot][\cdot]$ as follows: $v[i][c_j]$ is the number of votes that voter i has given to candidate c_j. Then, publishing the total number of votes $R_1, \ldots, R_K$ for each candidate gives rise to the following privacy goal:

$$\begin{aligned}\alpha \equiv \alpha_{ax'} \quad & \wedge v[1][c_1] + \ldots + v[1][c_K] = L \wedge \ldots \wedge v[N][c_1] + \ldots + v[N][c_K] = L \\ & \wedge v[1][c_1] + \ldots + v[N][c_1] = R_1 \wedge \ldots \wedge v[1][c_K] + \ldots + v[N][c_K] = R_K\end{aligned}$$

□

This shows that (α, β)-privacy indeed provides us with a new and declarative way to formulate privacy goals also in the context of e-voting, and sets the basis for the modeling and analysis of full voting protocols, which will also require us to consider (α, β)-privacy in the context of transition systems.

6 (α, β)-PRIVACY IN TRANSITION SYSTEMS

We now briefly show how we can extend (α, β)-privacy to transition systems. Given that transition systems are not the main focus of this paper (and will be investigated in detail in future work) we here only discuss the key idea and a detailed example of two security protocols for private authentication.

The key idea is that we can define a *state* as a triple (α, β, γ) of formulae, where γ represents the "truth". Then we can specify transition systems for this kind of states, and privacy is the question whether (α, β)-privacy holds in every reachable state (α, β, γ). Formally, with Σ, $\Sigma_0 \subseteq \Sigma$, $\mathcal{V}$ and $\approx$ as before:

Definition 6.1 (Transition systems). A *state* is a triple (α, β, γ), where α and β are as before and $\gamma \in \mathcal{L}_{\Sigma_0}(\mathcal{V})$ is such that $\gamma \models \alpha$ and γ is true in exactly one model of α (with respect to Σ_0 and the free variables of α). We also call γ the *truth* and may also apply it to Σ_0-terms like a substitution.

Let $\mathcal{S}$ denote the set of all states. A *transition system* is a pair (I, R) where $I \in \mathcal{S}$ and $R \subseteq \mathcal{S} \times \mathcal{S}$. As is standard, the set of *reachable states* is the smallest set that contains I and that is closed under R, i.e.: if S is reachable and $(S, S') \in R$, then also S' is reachable.

We say that a transition system *satisfies privacy* iff (α, β)-privacy holds in every reachable state (α, β, γ). □

Example 6.2. As an example of privacy as reachability, consider a simple transition system with an initial state that has no information, and four successor states $S_{i,j}$ with $i, j \in \{0, 1\}$ depending on two independent choices i and j of the user. In all four states, we have $\alpha \equiv x \in \{0, 1\}$. Let now $\beta_{i,j} \equiv \mathit{MsgAna}(\alpha, \mathit{struct}, \theta)$ where $\mathit{struct} = \{\!|\mathsf{m}_1 \mapsto \mathsf{scrypt}(\mathsf{k}_j, x), \mathsf{m}_2 \mapsto \mathsf{k}_1|\!\}$, k_j are new constants, and $\theta = \{x \mapsto i\}$.

In the states with $j = 0$, the intruder cannot deduce anything interesting as he does not have the key needed for decryption, but in the states with $j = 1$, we have $\beta_{i,1} \models x = i$. So, there are reachable states in which the intruder can find out more than he is supposed to. □

For an entire protocol one would not say that it satisfies its privacy goals if in every state (α, β)-privacy holds; one may then, by slight abuse of notation, say (α, β)-privacy holds for the protocol (referring to (α, β)-privacy as a concept, not with respect to a concrete pair of formulas α and β). That both α and β may grow over time should not be surprising, since the information that the intruder gathers may increase over time. For instance, at the beginning of a voting process, the result is not published yet (even if all voters have at this point already made up their mind how they want to vote). Moreover, a violation in an intermediate state of a system does not necessarily

entail a violation in the end (votes are actually released at the end but must be protected while the election is being carried out).

Let us now consider two more sophisticated examples.

6.1 Analysis of Two Example Protocols

To illustrate further the expressiveness and strength of (α, β)-privacy in transition systems, we consider two protocols proposed by Abadi and Fournet in **?** . The protocols are supposed to establish shared secrets between two agents A and B. As they write: "The first protocol uses digital signatures and requires that principals have loosely synchronized clocks. The second protocol uses only encryption and avoids the synchronization requirement, at the cost of an extra message. The second protocol draws attention to difficulties in achieving privacy against an active adversary." More information on the protocols can be found in [**?**]; here, we will focus only on what is needed to show (α, β)-privacy at work.

We adopt, whenever possible, Abadi and Fournet's notions and notations but introduce our own when needed. Consider a set of agents Agent and sets of agents S_A for every $A \in$ Agent, representing the set of agents that A is willing to talk to. In contrast to Abadi and Fournet, we write $\mathsf{pub}(A)$ and $\mathsf{priv}(A)$ for the public and private key of agent A, respectively.

6.1.1 The First Protocol (AF1). The first protocol, which we call AF1, is as follows:

$$A \to B : [\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(B), [\mathsf{hello}, \mathsf{pub}(A), \mathsf{sign}(\mathsf{priv}(A), [\mathsf{pub}(A), \mathsf{pub}(B), K, T])])]$$

where hello is a tag, K is a symmetric key freshly generated by A, T is a timestamp (we assume that B buffers all messages as long as their timestamp is considered recent, so replays can be detected) and $[t_1, \ldots, t_n]$ denotes the pairing $\mathsf{pair}(t_1, \mathsf{pair}(t_2, \mathsf{pair}(\ldots, t_n)))$.

Upon receipt of the message, the recipient B tries to decrypt the second component using its private key $\mathsf{priv}(B)$. If the decryption yields a key $\mathsf{pub}(A)$ and a signed statement of the form $\mathsf{sign}(\mathsf{priv}(A), [\mathsf{pub}(A), \mathsf{pub}(B), K, T])])$, then B extracts $\mathsf{pub}(A)$ and K, verifies the signature using $\mathsf{pub}(A)$, ensures that the message is not a replay using the timestamp T and checks that $A \in S_B$. If

- the form does not check out (it is not encrypted with $\mathsf{pub}(B)$, etc.),
- the message is a replay, or
- $A \notin S_B$ (i.e., B is not willing to talk to A),

then B should simply discard the message.

The encryptions are assumed to be *which-key concealing*, i.e., an intruder cannot tell which public key a message is encrypted with (unless he has the private key).

6.1.2 The Second Protocol (AF2). The second protocol, which we call AF2, consists of two steps:

$$\begin{aligned} A \to B :&\quad [\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(B), [\mathsf{hello}, N_A, \mathsf{pub}(A)])] \\ B \to A :&\quad \text{if valid then } [\mathsf{ack}, \mathsf{crypt}(\mathsf{pub}(A), [\mathsf{ack}, N_A, N_B, \mathsf{pub}(B)])] \text{ else } [\mathsf{ack}, R] \end{aligned}$$

where N_A and N_B are nonces and ack is an acknowledgment tag. We use an if-then-else to express that there is an error handling. B checks that the message (that apparently comes from A) is not a replay and tries to decrypt the second component using its private key. If the decryption succeeds, then B extracts the nonce N_A and key $pub(A)$, and checks that $A \in S_B$. If all these succeed, then the message is valid and B generates a nonce N_B, and sends a reply to A. If B receives an ill-formed message, then it will reply anyway, but with the second message in the "else" branch where R is a random value. Note that the original paper has here encryption of a new nonce with another public key of B, but the point is simply that without knowing $\mathsf{priv}(A)$, one should not be able to tell whether a given message was produced by the "then" or by the "else" branch. We thus assume

here something even slightly stronger than which-key-concealing: that also a random value is indistinguishable from the ciphertext, a property that at least holds in our algebraic model.[9]

Abadi and Fournet consider, among others, the following goal. If A wishes to communicate with B, but not vice versa, then the intruder should not learn anything. Thus, a run between the two agents A and B should be indistinguishable from a run between two other agents A' and B' under some hypotheses. These hypotheses should include that B is not the intruder and what can happen outside the protocol, i.e., what the agents can do besides running the protocol, which can result in leaks not caused by the protocol itself (see [**?**]).

6.1.3 Formalization of AF1 with (α, β)-privacy. In order to formalize AF1 with (α, β)-privacy, let the payload alphabet Σ_0 consist of

- a set of agent names (finite or countably infinite),
- a binary relation $talk(a, b)$ representing that a is willing to talk to b (in the notation of Abadi and Fournet: $b \in S_a$), and
- a unary predicate $honest(a)$ to identify a subset of agents a that are honest (and thus follow the protocol entirely); all other (dishonest) agents are just marionettes of the intruder.

In the previous sections, we have used a *substitution* θ to characterize an arbitrary model of α. Now we also have to interpret the predicates *honest* and *talk*, and therefore use a Σ_0-formula γ instead of θ. We will also use γ sometimes as a substitution; by construction we will ensure that whenever α in a state contains a variable x, then γ implies $x = a$ for some $a \in \Sigma_0$.

Definition 6.3. Let γ_0 be a closed Σ_0-formula with exactly one model (i.e., an arbitrary interpretation for the predicates *honest* and *talk*). The *initial state* $(\alpha_0, \beta_0, \gamma_0)$ of the transition system is as follows. Let $M_0 = \{a \mid a \in \Sigma_0\} \cup \{\mathsf{pub}(a) \mid a \in \Sigma_0\} \cup \{\mathsf{priv}(d) \mid \gamma_0 \not\models honest(d)\}$ be the set of ground messages initially known by the intruder. Let $struct_0 = \{\!|\mathsf{m}_1 \mapsto t_1, \ldots \mathsf{m}_l \mapsto t_l|\!\}$ if $t_1, \ldots, t_n$ are the elements of M_0. Define

$$\begin{aligned} \alpha_0 \equiv\ & \bigwedge\{talk(a,b) \mid \gamma_0 \models talk(a,b) \wedge \neg honest(a)\} \wedge \\ & \bigwedge\{\neg talk(a,b) \mid \gamma_0 \models \neg talk(a,b) \wedge \neg honest(a)\} \\ \beta \equiv\ & MsgAna(\alpha_0,\ struct_0,\ \gamma_0) \end{aligned}$$

Note that α_0 is ground. □

Observe that the initial state preserves (α, β)-privacy, since there are no variables, and $struct \sim \gamma_0(struct)$ in all models, thus it is consistent for all models of α.

There is only one kind of transition:

Definition 6.4. The *reachable states of AF1* are the least set of states that include the initial state and that are closed under the following transition rule. We use as an invariant that β in every reachable state is a message-analysis problem. If (α, β, γ) is a reachable state with $\beta = MsgAna(\alpha,\ struct,\ \gamma)$, then also the following state $(\alpha', \beta', \gamma')$ is reachable. Let a and b be any agents in Σ_0 such that $\gamma \models talk(a, b) \wedge honest(a)$. Let x and y be two fresh variables (that do not occur in α and β). Let k and t be arbitrary new uninterpreted constants of $\Sigma \setminus \Sigma_0$, and

$$\begin{aligned} struct' \ =\ & struct \cup \\ & \{\!|\mathsf{m}_{l+1} \mapsto [\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(y), [\mathsf{hello}, \mathsf{pub}(x), \mathsf{sign}(\mathsf{priv}(x), [\mathsf{pub}(x), \mathsf{pub}(y), k, t])])]|\!\}\,, \end{aligned}$$

[9] If one wants to model that ciphertexts are recognizable, one can simply introduce a new operator vciph with the algebraic property $\mathsf{vciph}(\mathsf{crypt}(k, m)) \approx \top$. Similarly one can model that a cipher is which-key-revealing with an operator vkey and the property $\mathsf{vkey}(k, \mathsf{crypt}(k, m)) \approx \top$.

where l is the length of the frame $struct$. Define

$$\begin{aligned}
\gamma' \equiv\ & \gamma \wedge x = a \wedge y = b \\
\alpha' \equiv\ & \alpha \wedge talk(x,y) \wedge \begin{cases} \bigwedge\{y \neq c \mid \gamma \models \neg honest(c)\} & \text{if } \gamma \models honest(b) \\ x = a \wedge y = b & \text{otherwise} \end{cases} \\
\beta' \equiv\ & MsgAna(\alpha',\ struct',\ \gamma')
\end{aligned}$$

□

Note that we describe only transitions that are made by honest agents (since the other agents are just intruder marionettes). That is, the intruder learns that there is an agent x who is willing to talk to y and has started a session; additionally, if y, i.e. b, is dishonest, then the intruder knows who x and y are; if y is honest, however, then the intruder only learns that y is none of the dishonest agents. In order to generate an (α, β)-privacy transition system we could, among other options, augment a process calculus notation in a suitable way. Then, to connect a process calculus notation with (α, β)-privacy here (for this transition) we would need to explicitly denote in the process calculus which information is released at this point, namely that $talk(x, y)$ is released to the public (i.e., to all) for arbitrary x and y, and $x = a$ and $y = b$ is released to y if x or y is dishonest. All the other information could be generated automatically in the process calculus notation.

Since (α, β)-privacy is not based on distinguishability, but rather a reachability problem, the proof of privacy is in fact a pretty straightforward induction proof:

Lemma 6.5. *Every reachable state of AF1 preserves (α, β)-privacy.*

Proof. First note that in no state the intruder learns the private key of an honest agent. The initial state preserves privacy as already noted. Let (α, β, γ) be any state in which privacy already holds and $(\alpha', \beta', \gamma')$ be any state that can be reached by one transition according to Definition 6.4. By the property of a reachable state, γ describes a single model of α. Moreover, x and y are exactly the new variables of α' and $talk(a, b)$ holds. Thus, γ' describes a single model of α'.

To see that β' is consistent for every model, we distinguish whether b is honest or not. If $\gamma \models honest(y)$, then the intruder does not know priv(b) (respectively priv(y)). Hence, the only check he can make (using the vcrypt function) is that the encrypted part cannot be decrypted with the private key of any dishonest agent, and thus that $y \neq c$ for any dishonest c. Since that is part of α', this cannot produce an inconsistency.

If $\gamma \models \neg honest(y)$, then α' already implies $x = a$ and $y = b$ and therefore $\beta' \models concr[l + 1] = struct[l + 1]$ where $concr = \gamma(struct)$. Thus, the addition to β' cannot produce an inconsistency either and (α, β)-privacy is preserved by every reachable state of AF1. □

6.1.4 Formalization of AF2 with (α, β)-privacy. For AF2, we have the same setup and initial state; only the transitions are different. The first transition rule is as follows.

Definition 6.6. If (α, β, γ) is a reachable state with $\beta = MsgAna(\alpha, struct, \gamma)$, then also the following state $(\alpha', \beta', \gamma')$ is reachable. Let a and b be any agents in Σ_0 such that $\gamma \models talk(a, b) \wedge honest(a)$. Let x and y be two fresh variables (that do not occur in α and β) and n_a be a fresh constant. Let

$$struct' = struct \cup \{\!|\mathsf{m}_{l+1} \mapsto [\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(y), [\mathsf{hello}, n_a, \mathsf{pub}(x)])]|\!\},$$

where l is the length of the frame *struct*. Define

$$\begin{aligned}\gamma' \equiv\ & \gamma \wedge x = a \wedge y = b\\ \alpha' \equiv\ & \alpha \wedge talk(x,y) \wedge \begin{cases}\bigwedge\{y \neq c \mid \gamma \models \neg honest(c)\} & \text{if } \gamma \models honest(b)\\ x = a \wedge y = b & \text{otherwise}\end{cases}\\ \beta' \equiv\ & MsgAna(\alpha',\ struct',\ \gamma')\end{aligned}$$

□

In a process calculus notation, here we would as before need to make explicit how α' extends α by the information that is released explicitly in this transition, namely $talk(x, y)$ is released to the public (i.e., to all) for arbitrary x and y, and $x = a$ and $y = b$ is released to y if y is dishonest, and otherwise, the intruder only learns that y is none of the dishonest agents. All the other information could be generated automatically in the process calculus notation.

One may argue that b cannot be sure at this point that it is really a who contacted it here. However, we do not model that several dishonest agents cheat each other, but that they all work together as the intruder. Hence, our model shall simply not include a dishonest a talking to a dishonest b trying to cheat about its identity. Thus, it is safe to assume that whenever b is dishonest and is apparently being contacted by a, it is indeed a (being either honest or an accomplice).

We now have a second transition where some message is received by an agent b and met with a reply (either the standard or the decoy message). We may assume that the message received by b has a proper form, namely

$$[\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(B), [\mathsf{hello}, N_A, \mathsf{pub}(A)])]$$

for some agents A and B and some term N_A. This message is either constructed by the intruder or a replay of a message the intruder has seen before. A priori, the intruder does not know whether the agent is actually b or somebody else (he can only guess; we assume the agent names to be all public). Thus, b will also answer with a decoy message if $B \neq b$ or if $\neg talk(B, A)$.

Definition 6.7. If (α, β, γ) is a reachable state with $\beta = MsgAna(\alpha,\ struct,\ \gamma)$, where l is the length of the frame *struct*, then also the following state $(\alpha', \beta', \gamma)$ is reachable. Let rp be any recipe such that

$$\beta \models gen(rp) \wedge struct[rp] = [\mathsf{hello}, \mathsf{crypt}(\mathsf{pub}(B), [\mathsf{hello}, N_A, \mathsf{pub}(A)])]$$

for some terms A, B and N_A, where A and B are either constants of Σ_0 or free variables of α. Let further $b \in \Sigma_0$ with $\gamma \models honest(b)$, and r and n_b be fresh constants (that do not appear in β). Then:

$$\begin{aligned}\alpha' \equiv\ & \alpha \wedge \begin{cases}B = b \wedge talk(B,A) & \text{if } \gamma \models B = b \wedge talk(B,A) \wedge \neg honest(A)\\ true & \text{otherwise}\end{cases}\\ \beta' \equiv\ & \exists s.\ MsgAna(\alpha,\ struct \cup \{\!|\mathsf{m}_{l+1} \mapsto s|\!\},\ \gamma' \cup \{s \mapsto t\})\\ & \quad \wedge (B = b \wedge\ talk(B,A) \iff s = [\mathsf{ack}, \mathsf{crypt}(\mathsf{pub}(A), [\mathsf{ack}, N_A, n_b, \mathsf{pub}(B)])])\\ & \quad \wedge (B \neq b \vee \neg talk(B,A) \iff s = [\mathsf{ack}, r])\\ t = & \begin{cases}[\mathsf{ack}, \mathsf{crypt}(\mathsf{pub}(\gamma(A)), [\mathsf{ack}, N_A, n_b, \mathsf{pub}(\gamma(B))])] & \text{if } \gamma \models B = b \wedge talk(B,A)\\ [\mathsf{ack}, r] & \text{otherwise}\end{cases}\end{aligned}$$

□

First note the difference in how the "case split" is handled: while the concrete message t is either the concrete answer message or the concrete decoy message (depending on the condition $B = b \wedge talk(B, A)$), for the structural information s we literally include the case split into the formula β, i.e., the intruder does not know the structure of the new message a priori, but he knows

that the structure is either the regular message (if the condition $B = b \wedge talk(B, A)$ holds true), or the decoy message, otherwise.[10]

The formula α' permits the intruder to learn that $B = b$ and $talk(B, A)$ if that is the case and A is a dishonest agent.[11] Otherwise, i.e., if $B \neq b$ or $\neg talk(B, A)$, the intruder shall learn nothing. The answer from b is thus either the normal message or the decoy message, depending on whether $B = b \wedge talk(B, A)$.

Observe that, if A is dishonest, then the intruder has composed the input message to B himself. Then the intruder knows $\mathsf{priv}(A)$ and can thus check whether the second part of the new message is (concretely) encrypted with $\mathsf{pub}(A)$. If that is the case, then it cannot be the decoy message, i.e., the intruder can derive $B = b \wedge talk(B, A)$ because of $\phi_{\Gamma_1 \sim \Gamma_2}$, but all this information is also part of α'.

If the decryption check fails, then it must be the decoy message and because of $\phi_{\Gamma_1 \sim \Gamma_2}$, $B \neq b \vee \neg talk(B, A)$. This formula does not follow from α' in general (since we can take this transition in the initial state) and we thus have a violation of (α, β)-privacy. Actually, we cannot guarantee that the A does not learn anything here; it must be that one of two things is the case: A has guessed wrongly who B is, or B does not want to talk to A. In fact, repeating this, A can guess a number of possible agent names and thus either find out who B is or find out (if an exhaustive search is possible) that $\neg talk(B, A)$.

As far as we can see the only way to correct this is to release this information in α':

Definition 6.8. Consider the second transition with this alternative α':

$$\alpha' \equiv \alpha \wedge \begin{cases} B = b \wedge talk(B, A) & \text{if } \gamma \models B = b \wedge talk(B, A) \wedge \neg honest(A) \\ B \neq b \vee \neg talk(B, A) & \text{if } \gamma \models (B \neq b \vee \neg talk(B, A)) \wedge \neg honest(A) \\ true & \text{otherwise} \end{cases}$$

□

Compare this with a different but similar situation: if the intruder tries to use online guessing for a login. We cannot entirely prevent it (we can only limit the number of passwords he can try within a given time) and also cannot prevent that with every failed attempt the intruder learns that the password he tried was not correct. With (α, β)-privacy we have a declarative way to express that: each failed guess leads to an augmentation of α.

Thus, (α, β)-privacy forces us to make explicit that we are leaking here a bit of information (which may be tolerable) and may lead to the awareness that we should protect the agent responding. In online guessing, it is common that after a failed attempt, one must wait a few seconds before a new attempt can be made. Similarly, here, B should pause a few seconds after any message. It is necessary to do so also in the successful case, since when A is honest it shall not be observable for the intruder whether a message was successful.

Lemma 6.9. *Every reachable state of AF2 with the modification of Definition 6.8 satisfies (α, β)-privacy.*

[10] Note that we do not specify transitions in Herbrand logic (just the states of the transition system are characterized by formulas in Herbrand logic), and so here we specified the transitions "by hand". Here we have modeled the case that the intruder uses a message for which he knows it has the right structure. This is clearly the case when he constructs such a message himself or if he uses one that was produced by an honest agent for the first step (because there is no decoy there). Sending any other message here necessarily would lead to a decoy and that is pointless for the intruder, so we omitted that here.

[11] In a process calculus notation, here we would need to make explicit how α' extends α by the information that is released explicitly in this transition, namely that the information $B = b$ and $talk(B, A)$ is released to the dishonest agent A.

Proof. Let (α, β, γ) a reachable state in which (α, β)-privacy holds. Let $(\alpha', \beta', \gamma')$ be state reached with one transition of Definition 6.6. First, γ' describes one single model of α'. Like in the first protocol, either b is honest or it is dishonest. If b is honest, then the intruder does not know $\mathrm{priv}(b)$ and cannot check the encrypted part of the message, and thus does not learn anything. If b is dishonest, then $\beta' \models \mathit{concr}[l+1] = \mathit{struct}[l+1]$ for the new position $l+1$ (where again $\mathit{concr} = \gamma(\mathit{struct})$) and thus it cannot lead to inconsistencies.

Let $(\alpha', \beta', \gamma')$ be a state reached with one transition of Definition 6.7 with the fix of Definition 6.8. We note again that γ' describes one single model of α'. If A is honest, then the intruder does not know $\mathrm{priv}(A)$ and thus cannot check whether the second part of the message at position $l+1$ is encrypted with $\mathrm{pub}(A)$ or is a decoy. Thus, the augmentation of β' is consistent. Otherwise, regardless of whether $\gamma \models B = b \wedge \mathit{talk}(B, A)$, we have $\beta' \models \mathit{struct}[l+1] = \mathit{concr}[l+1]$, and thus β' cannot introduce any inconsistencies with any model of α'. □

In future work, we will provide a more detailed account of (α, β)-privacy in transition systems, but we believe that these detailed examples already provide for a convincing case.

7 BACKGROUND KNOWLEDGE

One may wonder what happens if the intruder can make use of some background knowledge that is outside our formal model. Consider the following simplistic example. In a small rural village, everybody votes for the conservative party until one day a young couple moves to the village, and in the next election there are two left-wing votes. It does not take much imagination to figure out who was it (at least with high probability). Thus, with a bit of background information (and in fact common sense) one may be able to deduce information that was not deliberately released and in fact invade privacy. However note that this "attack" does not depend on the voting system: the best voting system cannot prevent this to happen since the system has been actually designed to release the total number of votes each candidate or party received. Of course, we cannot prevent an intruder from combining all the knowledge that is available to him, and thus, the voting system is not to blame as long as it does not release more information than specified in α.

The subtle question is however: given that we have verified a system to have (α, β)-privacy, but the intruder has some additional background knowledge α_0 that the formal model does not take into account, what guarantees do we have in the system then? Obviously, the intruder can derive anything that is implied by $\alpha \wedge \alpha_0$—the best system cannot prevent that. But could it be that the intruder can actually derive even more by some subtle combination of the information in β and α_0? We now show that this is not the case: our notion of (α, β)-privacy is *stable* under an arbitrary consistent intruder background knowledge, i.e., the intruder cannot derive more than $\alpha \wedge \alpha_0$:

Theorem 7.1. *Consider a pair (α, β) according to Definition 3.1 and let the intruder's* background knowledge *be any formula $\alpha_0 \in \mathcal{L}_{\Sigma_0}(fv(\alpha))$ such that $\beta \wedge \alpha_0$ is consistent. If (α, β)-privacy holds, then also $(\alpha \wedge \alpha_0, \beta \wedge \alpha_0)$-privacy holds.*

Proof. Suppose it were not the case, i.e., consider

- a pair (α, β) such that (α, β)-privacy holds,
- a background knowledge $\alpha_0 \in \mathcal{L}_{\Sigma_0}(fv(\alpha))$ such that $\beta \wedge \alpha_0$ is consistent, and
- an $\alpha' \in \mathcal{L}_{\Sigma_0}(fv(\alpha))$ such that $\beta \wedge \alpha_0 \models \alpha'$ but $\alpha \wedge \alpha_0 \not\models \alpha'$. (Thus, α' is a witness that $(\alpha \wedge \alpha_0, \beta \wedge \alpha_0)$-privacy does not hold.)

By Definition 3.1, β must have the form $\beta \equiv \alpha \wedge \beta_0$ for some β_0. We can thus rewrite $\beta \wedge \alpha_0 \models \alpha'$ as $\beta_0 \models \neg\alpha \vee \neg\alpha_0 \vee \alpha'$. Now

- $\beta_0 \models \neg\alpha$ is absurd, since then β would be inconsistent;
- $\beta_0 \models \neg\alpha_0$ is also absurd, since $\beta \wedge \alpha_0$ must be consistent; and

- $\beta_0 \models \alpha'$ would entail a violation of (α, β)-privacy as $\alpha \wedge \alpha_0 \not\models \alpha'$ and thus $\alpha \not\models \alpha'$. □

One may remark here that this also indicates why we get a reasonable model of privacy in quantitative systems without actually considering quantitative measures or probabilities: for a good voting system, for instance, it simply does not matter how likely the different outcomes are, the cryptography should treat them all the same, only the background knowledge may be biased, but that does not really matter for the system and its privacy properties then.

8 CONCLUDING REMARKS

We have introduced (α, β)-privacy as, we believe, a simple and declarative way to specify privacy goals and reason about them: the intruder should not be able to derive any "non-technical" statement from the technical information β that he cannot derive from the intentionally released information α already. We have given a variety of concrete examples that describe how (α, β)-privacy can be used in practice.

Above we have already compared extensively with static equivalence: we have described the simplicity of specifying properties via the declarative approach of (α, β)-privacy with respect to the more tricky specifications in the static equivalence approach, and we have investigated formally the close relationship of (α, β)-privacy to static equivalence, proving in particular the equivalence of message-analysis problems with a corresponding finite set of static equivalence problems. This result entails that we can use existing methods for deciding static equivalence for a given algebraic theory also for deciding the message-analysis fragment of (α, β)-privacy for that theory.

(α, β)-privacy is course also related to *observational equivalence* (see, e.g., [**?**] as well as works on *trace equivalence* [**? ? ? ?**]). In this approach, one typically considers labeled bi-similarity of two processes, checking that every transition of one process can be simulated by the other so that they are still bi-similar and so that their intruder knowledges are statically equivalent. This is difficult to automate but there are tricks that can be employed, such as turning it into a reachability problem by restricting the two processes to be the left and right variant of a bi-process. In contrast, thanks to the expressiveness of (α, β)-privacy, we have a way to formulate privacy as a reachability problem while not being limited by bi-processes and the like. We believe that this gives opportunities for novel techniques for the automated verification of privacy goals.

(α, β)-privacy bears some similarities also with the *non-interference* approach and related works in information-flow and language-based security (see, e.g., [**? ? ? ?**]). Non-interference distinguishes (at least) two levels of information, usually low-level and high-variables. These are, however, fundamentally different from our payload α and technical information β since they are formulae that express relations between values (rather than directly being public or private values). We actually do not mind that the intruder gets hold of (some) technical information as long as he cannot use it to obtain anything interesting besides the payload.

Privacy has also been studied in the area of statistical databases, building on database abstractions, in which records may contain identifiers, quasi-identifiers and sensitive attributes. Approaches in this field (such as k-anonymity, ℓ-diversity, t-closeness and differential privacy) aim at quantifying privacy in order to capture privacy loss and thus analyze the minimal information disclosure inherent in a system. *k-anonymity* [**? ?**] seeks to protect against identity disclosure by ensuring that a record is indistinguishable from $k - 1$ other records on quasi-identifiers, usually replaying quasi-identifiers with equivalence classes of appropriate size. Hence, an intruder must be unable to reduce the anonymity set below a threshold of k users. It is known that k-anonymity does not protect against attribute disclosure, which led to the development of *ℓ-diversity* [**?**], i.e., the requirement that each equivalence class contains at least ℓ representations of a sensitive attribute. The work on *t-closeness* [**?**] observed that there are still attribute disclosures possible in ℓ-diverse

datasets, in particular, when the distribution of a sensitive attribute in an equivalence class is different from its distribution in the whole database, where this new notion stipulates that the distance between equivalence class and table distribution is at most t. Let us differentiate which statements over these three notions can be modeled in (α, β)-privacy.

For k-anonymity, we observe that the property that α has at least k models, and that the intruder cannot deduce an α' with less choices, is in principle encodable in (α, β)-privacy, which means that (α, β)-privacy can express identity disclosure, although a full-fledged encoding of k-anonymity in (α, β)-privacy will be subject of future work.

Our argument can be applied recursively to equivalence classes: for each equivalence class for any sensitive attribute, there are ℓ models, and the intruder cannot deduce an α' with less choices for any equivalence class. Hence, the argument suggests also an (α, β)-privacy encoding for the main definition of distinct ℓ-diversity. However, (α, β)-privacy does not have a notion of entropy and thus cannot encode entropy ℓ-diversity.

Similarly, (α, β)-privacy cannot encode directly t-closeness, which relies on distance of probability distributions (using variational distance or the Kuhlback-Leibler distance on entropies as measure).

Finally, let us consider *differential privacy* [**?**], which asks whether an intruder can detect significant changes in a probability distribution on statistical data released by a curator on data sets differing in one element. As differential privacy is a property established on the information release function of the curator, a relation to our notion is not straightforward.

We have already mentioned above and in the previous sections a few directions for future work. In addition to these, we have already started to consider further examples than those discussed here, in particular examples that fall outside the message-analysis problem. To that end, we will need to generalize the definition of combinatoric α and to generalize our decidability results to larger fragments of (α, β)-privacy. In fact, many interesting issues in e-voting fall outside the message-analysis fragment (and of the static equivalence approach).

We also plan to extend our formalization to a full-fledged specification of (α, β)-privacy in transition systems. It will also be interesting to investigate how (α, β)-privacy, which is a purely qualitative and possibilistic approach, can be extended to consider quantitative aspects of privacy such as: probabilities, time and cost of the private information.

ACKNOWLEDGMENTS

We thank Thomas Groß for our joint preliminary work and for many interesting discussions. This work was supported by the Sapere-Aude project "Composec: Secure Composition of Distributed Systems", grant 4184-00334B of the Danish Council for Independent Research; the EU FP7 project no. 318424, "FutureID: Shaping the Future of Electronic Identity" (futureid.eu); the EU FP7 project no. 257876, "SPaCIoS: Secure Provision and Consumption in the Internet of Services" (www.spacios.eu); the EU H2020 project no. 700321 "LIGHTest: Lightweight Infrastructure for Global Heterogeneous Trust management in support of an open Ecosystem of Trust schemes" (lightest.eu); and the Italian PRIN 2010-11 project "Security Horizons".

Received December 2017; revised July 2018; accepted October 2018